UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE INFORMATIQUE PARIS-SUD
Laboratoire de CEA Tech LIST

DISCIPLINE : Informatique

THÈSE DE DOCTORAT

soutenue le 07/07/2014

par

Ernest Wozniak

Model-based Synthesis of Distributed Real-time Automotive Architectures

Directeur de thèse:	Dr. Sébastien Gérard	CEA Tech LIST
Co-directeur de thèse:	Dr. Chokri Mraidha	CEA Tech LIST

Composition du jury

Président du jury :	Burkhart Wolff	Professeur (Université Paris-Sud, CNRS)
Rapporteur :	Maryline Chetto	Professeur (Université de Nantes, IRCCyN Research Institute)
Rapporteur :	Martin Törngren	Professeur (KTH Royal Institute of Technology)
Examinateur :	Claire Pagetti	Ingénieur-Chercheur (ONERA)
Examinateur :	Manfred Broy	Professeur (Technische Universität München)

Model-based Synthesis of Distributed Real-time Automotive Architectures

ACKNOWLEDGEMENTS

First I wish to thank my supervisor Dr. Sébastien Gérard. He guided me into the domain of modeling and provided almost unlimited resources and support for my research. It was a great pleasure for me to work with Sébastien Gérard. His professionalism, friendliness and openness to other people let me to simply enjoy my work under his supervision.

I am very grateful to my co-supervisor, Dr. Chokri Mraidha for his guidance throughout my entire PhD work from the very beginning where he helped me to arrange my arrival to the CEA Tech LIST institute till the very end where he dealt with all the necessary preparations to organize my defense. I am especially thankful for all his reviews done on my papers and thesis report as well as numerous discussions that influenced my research.

I would like to thank to Dr. Sara Tucci-Piergiovanni for all the discussions that we had. They were very motivating and had a great impact on the direction of my research. Her academic skills which I truly admire instilled in me a real scientific spirit.

I am deeply grateful to Prof. Haibo Zeng for hosting me as an exchange PhD student during my stay at the McGill University. Work with him was a lifetime experience and a great privilege which cemented and further expanded my knowledge.

I also had a great pleasure to work with Prof. Marco Di Natale whom I would like to thank for the numerous, fruitful, remote discussions and his commitment in a work on publications that I had a pleasure to publish with him.

I am grateful to my reviewers, i.e. Prof. Maryline Chetto and Prof. Martin Törngren for agreeing to evaluate my work and for their insightful comments. I am also very thankful to them for devoting their time to participate in my defense and be part of the committee. Similarly I would like to thank other committee members, i.e. Prof. Claire Pagetti, Prof. Burkhart Wolff and Prof. Manfred Broy. It was a privilege to have you in my jury.

I am thankful to all of my colleagues at the CEA Tech LIST for a fabulous working environment, discussions and their patience in understanding and helping to improve my

French. I especially thank to Asma Mehiaoui, Ahmed Daghsen and Florian Noyrit for the joint work, conversations and sharing an office with me.

I am very grateful to my cousin Marcin Fedor who showed me numerous opportunities for professional development and inspired me to pursue the PhD work.

Last but not least I would like to thank my parents. They were always very supportive in this fabulous PhD journey. Dziękuję Wam drodzy rodzice za wsparcie!

PUBLICATIONS

- **E. Wozniak**, M. D. Natale, H. Zeng, C. Mraidha, S. T. Piergiovanni, and S. Gérard, „Assigning Time Budgets to Component Functions in the Design of Time-Critical Automotive Systems," *Automated Software Engineering (ASE), 2014 IEEE/ACM 29th International Conference, Västerås, Sweden, September 15-19, 2014*
- **E. Wozniak**, C. Mraidha, S. Tucci-Piergiovanni, S. Gerard, "An Optimization Approach for the Synthesis of AUTOSAR Architectures," in Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Cagliari, Italy, September 10-13, 2013.
- A. Mehiaoui, **E. Wozniak**, S. T. Piergiovanni, C. Mraidha, M. D. Natale, H. Zeng, J.-P. Babau, L. Lemarchand, and S. Gérard, "A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems," in *LCTES*, 2013, Seattle, WA, USA, pp. 121–132.
- **E. Wozniak**, C. Mraidha, S. Tucci-Piergiovanni, S. Gerard "A Model-Based Approach for Real-Time Systems Architecture Exploration", *IST-115 Symposium on Architecture Definition & Evaluation (The NATO Science and Technology Organization), Toulouse, France 13 - 14 MAY 2013*
- **E. Wozniak**, S. Tucci-Piergiovanni, C. Mraidha, and S. Gerard, "An Integrated Approach for Modeling, Analysis and Optimization of Systems whose Design Follows the EASTADL2/ AUTOSAR Methodology," SAE International Journal of Passenger Cars - Electronic and Electrical Systems, May 2013 vol. 6 no. 1 276-286
- **E. Wozniak**, C. Mraidha, S. Gerard, "Guided Task Model Construction for Automotive Systems based on Time Budgets", *in Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2012), Cracow, Poland, 2012*
- S. Tucci-Piergiovanni, C. Mraidha, **E. Wozniak**, A. Lanusse, and S. Gerard „A UML Model-based Approach for Replication Assessment of AUTOSAR Safety-critical Applications", *In Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (Changsha, China).*
- **E. Wozniak**, C. Mraidha, S. Gerard, F. Terrier, "Guidance Framework for the Generation of Implementation Models in the Automotive Domain", *in Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA 2011) (Oulu, Finland).*

ABSTRACT

Model-based Synthesis of Distributed Real-time Automotive Architectures

by

Ernest Wozniak

Hardware/software based solutions play significant role in the automotive domain. They deliver functionality that normally wouldn’t be accomplishable with pure mechanics or electronics. In fact it is common that the implementation of certain functions that was done in a mechanical manner, like hydraulic brakes, in nowadays cars is done through the software and hardware. This tendency lead to the substantial number of functions operating as a set of software components deployed into hardware entities, i.e. Electronic Control Units (ECU). As a consequence the capacity of the overall code is estimated as tens of gigabytes and the number of ECUs easily reaches 50 up to 80. Therefore the industrial state of the practice development approaches become inefficient.

The objective of this thesis is to add to the current efforts trying to employ the Model Driven Engineering (MDE) in the context of the automotive SW/HW architectures design. Adoption of the MDE is a sound choice towards an efficient and cheaper development process. To comply with this tendency this work introduces a framework developed as an instance of an Architecture Framework and aligned to the principles of the MDE. It serves for the modeling, analysis and optimization of the automotive architectures. Within the context of this framework a set of particular contributions is presented.

First set of contributions relates to the guided strategies supporting the key engineering activities of the EAST-ADL2/AUTOSAR methodology. The main is the integration of the software architecture with the hardware platform. Although the amount of work on the synthesis is substantial, this thesis presents shortcomings that disable them to fully support the EAST-ADL2/AUTOSAR methodology. Firstly, this is

the omission of functional entities and secondly, incapability to handle situations in which execution times for them are missing. Presence of the functional entities is due to the introduction of the atomic functions and runnable entities correspondingly in the EAST-ADL2 and the AUTOSAR. The missing execution times, this is a very likely scenario to occur during the integration process as the synthesis is done on the abstract models without code implementation which is necessary to estimate them. The presence of the lasts is obligatory to enable qualitative synthesis. On the canvas of these shortcomings, a collection of new techniques to handle the synthesis step is presented. They account for the functional entities, are capable of dealing with missing execution times and optimize key parameters of the architectures, i.e. the end-to-end responses and the memory.

Second contribution concerns approaches for the UML based modeling. Comprehensible specification is the key factor for the effective maintenance of the system architecture throughout the development cycle. Surprisingly the usage of general purpose modeling languages such as the UML, SysML and MARTE although beneficial, haven’t found its way yet to be fully exploited by the automotive OEMs (Original Equipment Manufacturer). This especially relates to the modeling of the analyzable input and the optimization concerns which would enable the analysis and optimization to be run directly on the model or generation of the input models for the other tools that serve for this purpose. Consequently this thesis presents models supporting these concerns, expressed with the OMG (Object Management Group) standards: UML, SysML and MARTE.

RESUME

Synthèse Basée sur les Modèles d'Architectures Automobiles Temps Réel Distribuées

1. Contexte de la thèse

Les systèmes véhicules d'aujourd'hui sont caractérisés par une large gamme de solutions qui améliorent la performance, la sécurité et le confort de conduite. Des fonctionnalités telles que le système stationnement automatique vont au-delà des attentes des conducteurs ordinaires d'il y a seulement 10 ans. Les premiers prototypes des véhicules autonomes ont déjà été réalisés.

Les systèmes automobiles sont des systèmes distribués, embarqués et tems réel. Tout d'abord, les fonctionnalités logicielles des véhicules sont distribuées sur plusieurs composants matériels embarqués nommées unités de commande électronique (ang. ECU – Electronic Control Unit) ou sur des capteurs/actionneurs. La couche d'application qui s'étend sur des ECU différents est composée de composants logiciels qui peuvent être conçus et livrés par plusieurs fournisseurs. Le middleware est responsable de la communication entre les composants logiciels distribués. Chaque ECU exécute un système d'exploitation. Tout cela implique une nature distribuée des systèmes automobiles (voir la Figure 1). Deuxièmement leur fonctionnement est contraint par des contraintes de temps de différents types, par exemple des contraintes temporelles de bout-en-bout. Par exemple, l'ouverture de l'airbag en cas d'accident doit se produire dans les 20 ms. Cette dernière est une contrainte de temps réel, dont la violation non seulement affirme le comportement incorrect du système, mais plus important, peut mettre en danger la vie de personnes humaines.

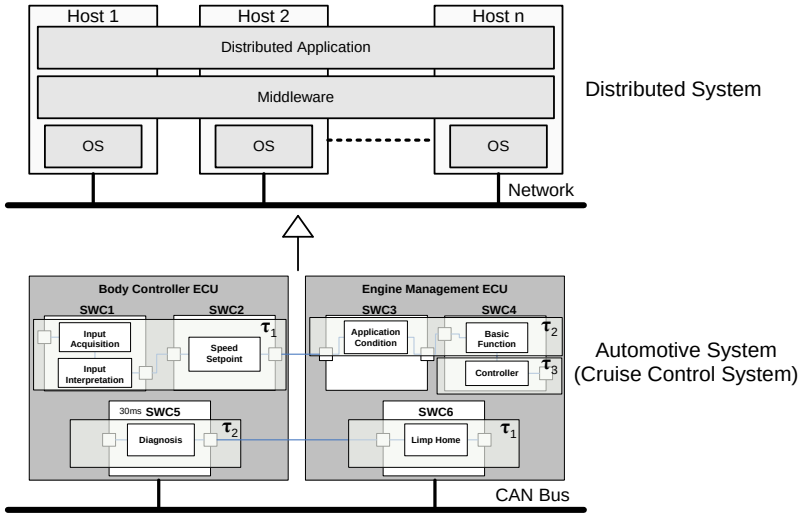

Figure 1. Système Distribué et Système Distribué Automobile

Les architectures de systèmes automobiles (en raccourci architectures automobiles) sont des produits très complexes, de haute technologie. Différents facteurs contribuent à leur complexité:

- **Taille:** le nombre de fonctions contrôlées par le logiciel et le matériel est substantiel dans les véhicules d'aujourd'hui. En une trentaine d'années, la quantité de code est passée de 0 à près de 10 Go, ce qui représente des millions de lignes de code.
- **Nature distribuée:** les architectures automobiles d'aujourd'hui sont fortement distribuées, c.-à-d. les fonctions atomiques de la même fonctionnalités d'un véhicule sont distribuées sur plusieurs ECUs. Le même ECU peut accueillir des fonctions atomiques de différentes fonctionnalités du véhicule. Cela permet une meilleure optimisation de l'utilisation des ressources.
- **Les contraintes temps réel:** le fonctionnement correct d'un système de véhicule n'est pas seulement défini par l'absence d'erreurs fonctionnelles, mais aussi par strict respect des contraintes temps réel. Leur existence sert principalement dans les situations critiques pour la sécurité, comme le freinage ou pendant un accident lorsque les airbags doivent être activés immédiatement.
- **Exigences de sécurité:** l'aspect de la sécurité joue un rôle important car maintenant ce n'est pas seulement une préoccupation interne d'un OEM (ang.

Original Equipment Manufacturer) de fournir des véhicules fiables, mais aussi un sujet pour les réglementations gouvernementales.

- **Exigences contradictoires:** les différentes exigences comme les contraintes de temps, la réduction des ressources matérielles pour réduire les coûts, la sécurité, etc. sont dans de nombreux cas des exigences orthogonales. Cela signifie que la satisfaction d'une exigence peut conduire la dégradation des autres exigences.
- **Sensible aux changements:** de légers changements de conception ou certaines propriétés des éléments d'architecture peuvent conduire à une modification radicale des caractéristiques non-fonctionnelles d'architecture. Par exemple, l'augmentation d'un temps d'exécution d'une fonction atomique peut conduire à la violation de plusieurs contraintes de temps.

En raison de cette complexité qui a été et est encore en croissance exponentielle (prévue pour les 20 prochaines années), de nouvelles stratégies pour la conception des systèmes automobiles doivent être introduites. L'une d'entre elles est l'adoption de l'ingénierie dirigée par les modèles (IDM) pour le développement des systèmes automobiles. Le principe de l'approche IDM consiste à intégrer des modèles pour spécifier les exigences fonctionnelles et non fonctionnelles, et enfin, pour produire un code binaire qui respecte la spécification. Le potentiel de l'IDM a été identifié par les grands constructeurs automobiles et les fournisseurs qui ont initié un projet avec un objectif de fournir un standard commun fondé sur les principes de l'IDM. Ce projet appelé AUTOSAR (Automotive Open System Architecture) est actuellement le standard la plus influent en termes de modélisation des systèmes automobiles. La chaîne de développement de la méthodologie AUTOSAR (voir Figure 2) s'étend à partir de la représentation de composants logiciels d'application à l'infrastructure d'exécution, y compris la description de la plate-forme matérielle. Un inconvénient d'AUTOSAR est son manque de support pour la modélisation du niveau fonctionnel. Par conséquent, il y a un intérêt dans la combinaison de ce standard avec le langage de modélisation EAST-ADL2 qui prend en charge la spécification fonctionnelle.

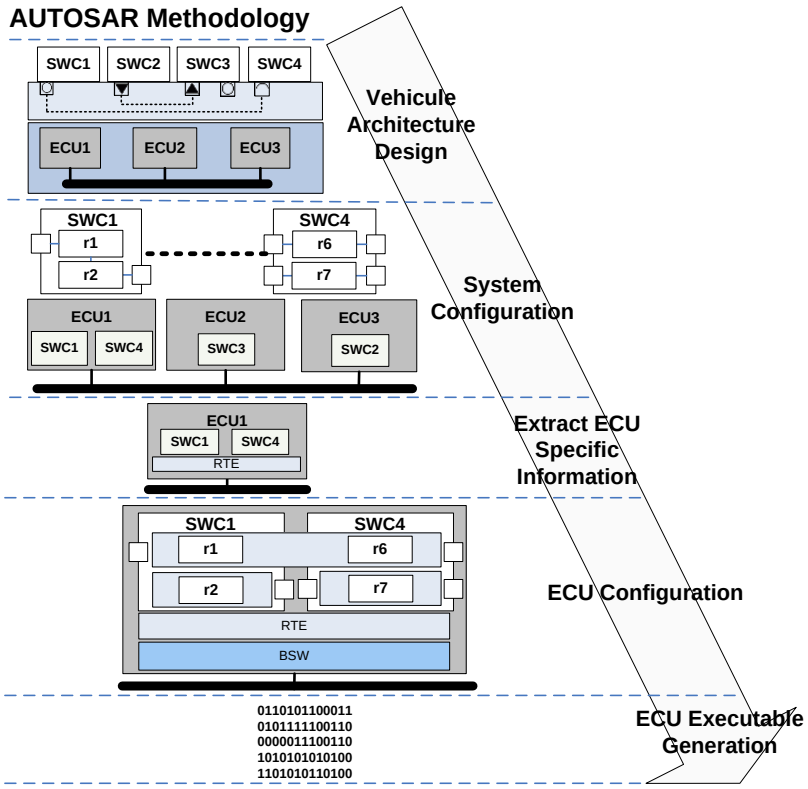

Figure 2. AUTOSAR Méthodologie

EAST-ADL2 et AUTOSAR imposent des règles méthodologiques pour la construction de modèles. Leur avantage est qu'ils fournissent un cadre commun pour la conception de systèmes électroniques automobile. Toutefois, aucune de ces méthodologies ne définit comment effectuer certaines étapes de conception, par exemple, la façon de distribuer les composants logiciels sur les éléments matériels ou comment partitionner des entités fonctionnelles sur des tâches OS (ang. Operating System). À cet égard, ces deux standards comptent entièrement sur une expérience de concepteur, augmentant ainsi le potentiel nombre de défauts de conception. En conséquence, il est essentiel de procéder à une analyse comme, l'analyse temporelle ou l'analyse de sécurité pour assurer que les décisions prises par le concepteur n'a pas conduit à des architectures irréalisables. Nous pouvons aller encore plus loin et utiliser des techniques pour l'exploration de l'espace de conception (ang. DSE – Design Space Exploration). Leur emploi pourrait assurer la faisabilité, mais en plus permet d'optimiser les propriétés non-fonctionnelles clés.

2. Énoncé du problème & motivation

Comme indiqué dans le paragraphe précédent, une vision claire et exhaustive d'une conception de système automobile, ainsi que son analyse / optimisation, sont les activités nécessaires pour rester compétitif sur le marché de l'automobile. Cela nécessite des langages de modélisation, de méthodes d'analyse et des techniques pour permettre le DSE. L'objectif général et initial de cette thèse est d'intégrer ces trois activités dans un cadre méthodologique, soutien de la conception des architectures automobiles et suivie par la méthodologie EAST-ADL2/AUTOSAR. Dans ce cadre, un ensemble de problèmes intéressants sont posés. La recherche de solutions appropriées est important pour rendre possible l'intégration de ces activités et la fourniture d'un flot continu guidé entre eux pour finalement produire un modèle d'implémentation optimisé d'un système automobile.

Disposer de différents types de modèles, c.-à-d. le modèle d'architecture, modèle d'analyse et modèle d'optimisation, est nécessaire pour effectuer une synthèse optimisée du logiciel avec le matériel. La phase principale de la synthèse est appelée déploiement. Selon AUTOSAR, le déploiement concerne 1) **l'**allocation des composants logiciels sur ECU 2) le partitionnement des entités du comportement du composant (appelées runnable entities) sur des OS tâches et enfin 3) l'ordonnancement des tâches OS. Un point crucial pour cette étape est sa validité en fonction de ses propriétés temporelles. Depuis le raffinement du système (dont le déploiement est une partie intégrante), la validité peut être assurée sous certaines hypothèses concernant des détails de niveau inférieur. Un exemple typique est l'hypothèse sur la connaissance des temps d'exécution pire cas (WCETs) des entités exécutables AUTOSAR. Il est évident que l'hypothèse de la connaissance précise des WCETs de runnables avant l'implémentation du code est la plupart du temps irréaliste**.** Dans de nombreux cas, certains runnables de systèmes précédents sont réutilisés. Le WCET de ces runnables est alors connu. Cependant, ce n'est pas le cas quand les nouveaux runnables implémentant de nouvelles fonctionnalités sont introduits. Cela représente un problème pour la synthèse de l'architecture et, en général, la fourniture d'un flot top-down**.** Ce qui est encore plus important est que le déploiement défini dans AUTOSAR n'est pas supporté de manière holistique par les techniques existantes. Bien que la quantité de travail qui existe semble être conséquente,

un fossé existe**.** Les techniques proposées soit représentent les tâches OS comme entités d'allocation ou résolvent le problème dans les étapes sans tenir compte d'un impact négatif qu'elle a sur les résultats finaux par rapport à l'approche holistique**.**

EAST-ADL2 et les spécifications AUTOSAR offrent un large éventail de concepts qui sont nécessaires pour définir l'architecture complète d'un système. Les efforts récents pour étendre ces standards ont fourni les capacités pour modéliser les informations nécessaires à l'analyse temporelle. Le travail adéquat n'a pas été fait jusqu'à présent pour gérer les optimisations. Bien que le domaine temps réel et des systèmes distribués est riche en techniques d'optimisation**,** il n'y a pas de concepts de modélisation qui permettraient de spécifier une entrée nécessaire pour cette activité tels que les objectifs d'optimisation (temps des réponses, la consommation de mémoire, etc.). En conséquence, la modélisation et l'analyse/l'optimisation ne sont pas bien intégrées. Cela a abouti à de nombreux outils décousus pour la modélisation ou l'analyse et/ou l'optimisation.

3. Contributions

Afin de permettre de développement sans couture dans le cadre proposé, ce travail propose un ensemble de solutions aux problèmes mentionnés ci-dessus:

1) Concernant les techniques de DES les principales contributions portent sur la définition de nouvelles techniques pour optimiser le déploiement. Les techniques proposées sont conformes à la définition du déploiement comme inclus dans le standard AUTOSAR. C'est-à-dire, ils considèrent les runnable entities que les unités d'allocation. Par conséquent, l'étape de partitionnement qui n'est pas considérée par les approches existantes est supportée par la technique définie dans ce travail. Les techniques proposées sont basées sur des heuristiques, algorithmes évolutionnistes, diviser pour régner, amélioration itérative, c'est pourquoi ils sont capables de traiter de grandes architectures d'entrée. Cette caractéristique a été évaluée en effectuant plusieurs tests, atteignant 250 runnables. Un critère d'évaluation important a été la qualité des architectures déployées. Ceci a été réalisé en comparant les résultats à ceux obtenus avec les méthodes exactes ou des architectures pour lesquelles la configuration optimale de déploiement était connue a priori. Dans AUTOSAR, des modèles de comportement

pourraient correspondre soit à une sémantique d'exécution data driven ou time driven. Cela nécessite de définir les différents types de stratégies d'optimisation. La différence se situe dans l'analyse d'ordonnançabilité. En outre, les métriques d'optimisation telles que les métriques temporelles, les métriques de mémoire sont affectées d'une manière différente par les choix particuliers d'un déploiement. Ce qui caractérise aussi les techniques proposées est la prise en compte de critères multiples (par exemple, les réponses de bout-en-bout, les propriétés temporelles, la consommation mémoire) qui définit une bonne configuration de déploiement de l'architecture d'entrée. La Figure 3 montre un exemple de l'architecture logicielle d'entrée (partie supérieure) et sa spécification de déploiement.

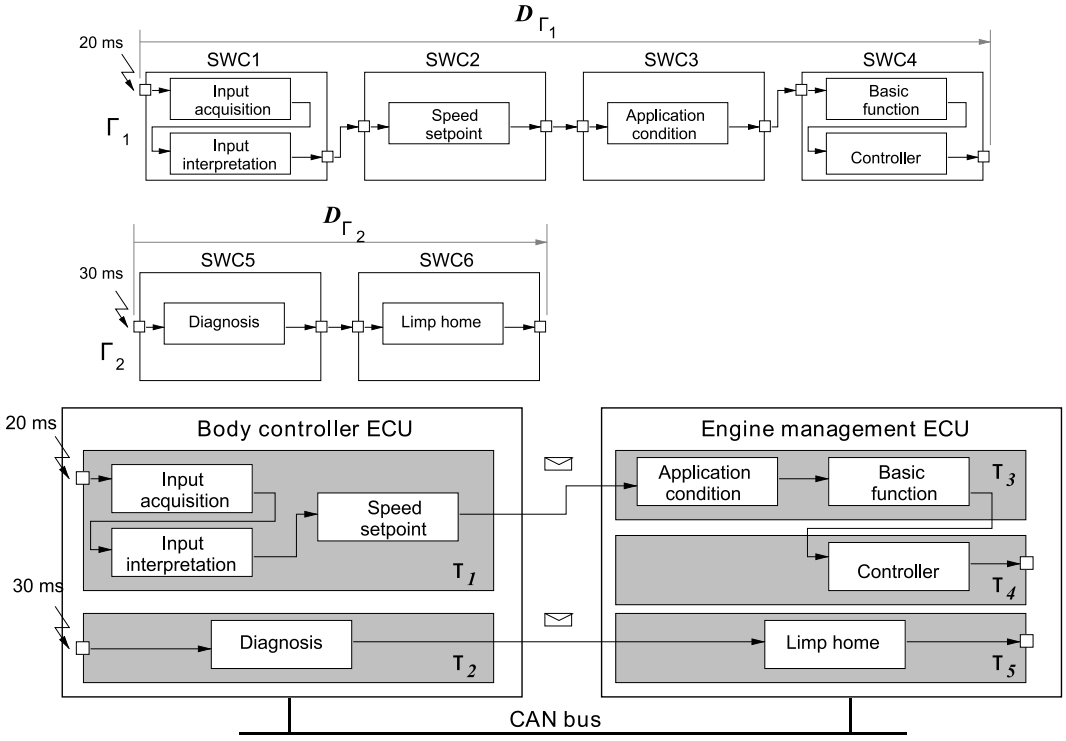

Figure 3. Exemple de l'architecture logicielle d'entrée (partie supérieure) et sa spécification de déploiement (partie basse)

.

2) Pour améliorer les résultats d'un déploiement, ce travail suggère un raffinement de la méthodologie EST-ADL2/AUTOSAR. Le but est de permettre de résoudre de manière holistique le problème de déploiement, ce qui n'est pas possible avec définition

actuelle de cette méthodologie. Le changement concerne la répartition des responsabilités entre les deux niveaux, le niveau fonctionnel couvert par EST-ADL2 et le niveau implémentation couvert par AUTOSAR. L'activité *Design* qui se fait au niveau fonctionnel comprend l'étape d'allocation des fonctions atomiques aux ressources matérielles, c.-à-d. ECUs. Ceci détermine la répartition des runnable entities en raison de l'hypothèse dans laquelle les runnable entities sont transformés à partir des fonctions atomiques. C'est pourquoi le problème de déploiement ne peut pas être résolu d'une manière holistique au niveau AUTOSAR parce qu'une dimension du problème, c.-à-d. l'allocation est déjà fixée. Par conséquent, ce travail préconise le changement dans lequel l'allocation est reportée jusqu'au niveau d'implémentation. L'évaluation de ce changement montré une amélioration remarquable des caractéristiques de l'architecture.

3) Pour effectuer un déploiement qui optimise les réponses de bout-en-bout, les temps d'exécution des runnable entities sont nécessaires. Comme cette information peut-être manquante pour certains runnables, la définition d'une nouvelle stratégie pour la configuration de l'architecture est inévitable. Pour contourner le problème, certains travaux proposent d'ajouter à la méthodologie d'une activité appelée budgétisation de temps (ang. time budgeting). Au lieu d'estimer WCETs, l'intégrateur de système spécifie des budgets temporels (ang. time budgets), c'est à dire des contraintes à des temps de réponse pire cas - WCRTs (ang. Worst Case Execution Times). Les budgets temporels doivent être respectés par les fournisseurs livrant l'implémentation des composants. Le problème typique de cette approche est que le fournisseur livre l'implémentation d'un composant particulier, qui sera intégrée par l'intégrateur en tant que partie intégrante du système, dans une étape ultérieure. Entre temps, le fournisseur valide le composant en isolation, sans tenir compte d'éventuelles interférences avec d'autres composants. Il est alors incapable de calculer un temps de réponse pire cas (WCRT) correct. C'est-à-dire si le composant répond à la contrainte du budget temporel l'intégrateur du système doit prendre soin d'éviter toute interférence possible avec d'autres composants. Ce n'est pas seulement une tâche difficile, mais qui provoque généralement un surdimensionnement des ressources. Ce surdimensionnement des ressources peut représenter des coûts insoutenables pour une production en série. Une solution alternative est celle dans laquelle les budgets temporels représentent des contraintes de WCET de runnable entity

à la place de son WCRT. Ce travail propose une solution de budgétisation des WCETs. La plupart des travaux existants sur budget de WCRT ne pas bien adaptés à l'idée de « l'architecture intégré » proposée par AUTOSAR. Un autre avantage de la technique proposée dans ce travail par rapport aux approches existantes est l'hypothèse que le déploiement n'est pas connu à l'avance. En conséquence, un objectif de la technique proposée est de trouver conjointement le déploiement et l'affectation optimale des budgets temporels. La Figure 4 illustre une architecture logicielle d'entrée et une architecture matérielle pour lesquelles le déploiement ainsi que les budgets de temporels doit être spécifié. En fait, les budgets temporels doivent être définis pour ces runnable entities pour lesquels les informations sur le WCET n'est pas présent. La Figure 5 présente le résultat de la technique proposée, c.-à-d. l'architecture déployée ainsi que les budgets temporels.

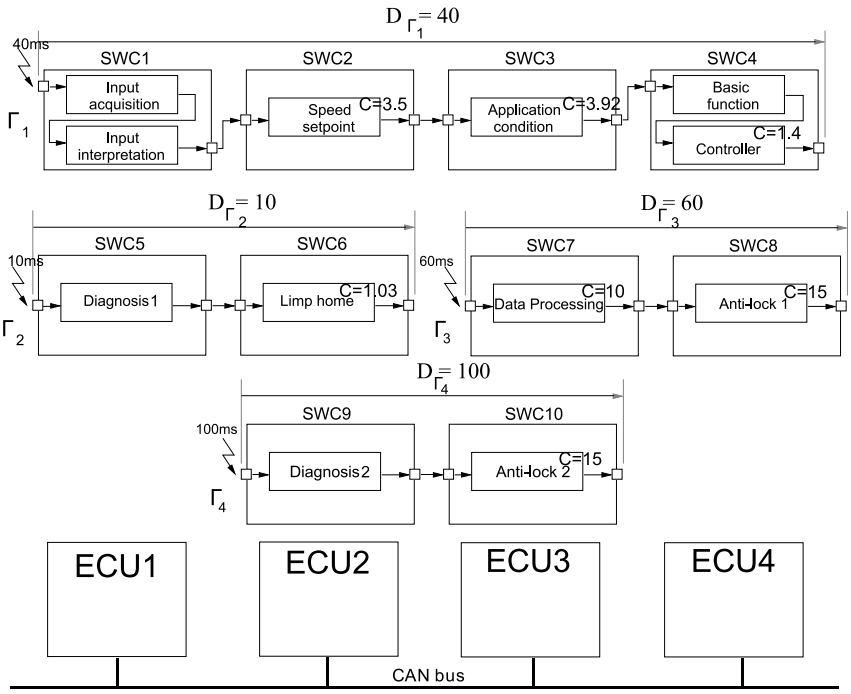

Figure 4. L'architecture Logicielle et Matérielle avec certains Runnables qui manquent WCETs

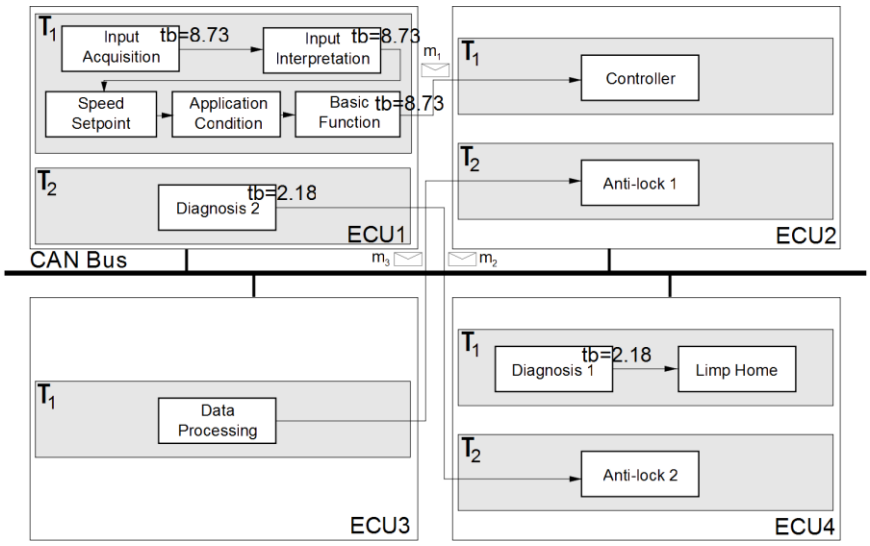

Figure 5. Résultant Déploiement avec la Spécification des Budgets Temps

4) En ce qui concerne la modélisation de la première contribution, une spécification de concepts essentiels pour construire un modèle d'optimisation et pour exécuter des techniques DES telles que celles définies dans ce travail, servant pour le déploiement ou budgétisation de temps, a été réalisée. La Figure 6 représente une partie du profil UML définissant les principaux concepts permettant de construire un contexte d'optimisation. Au-dessus du modèle d'optimisation, des techniques d'optimisation peuvent être exécutés.

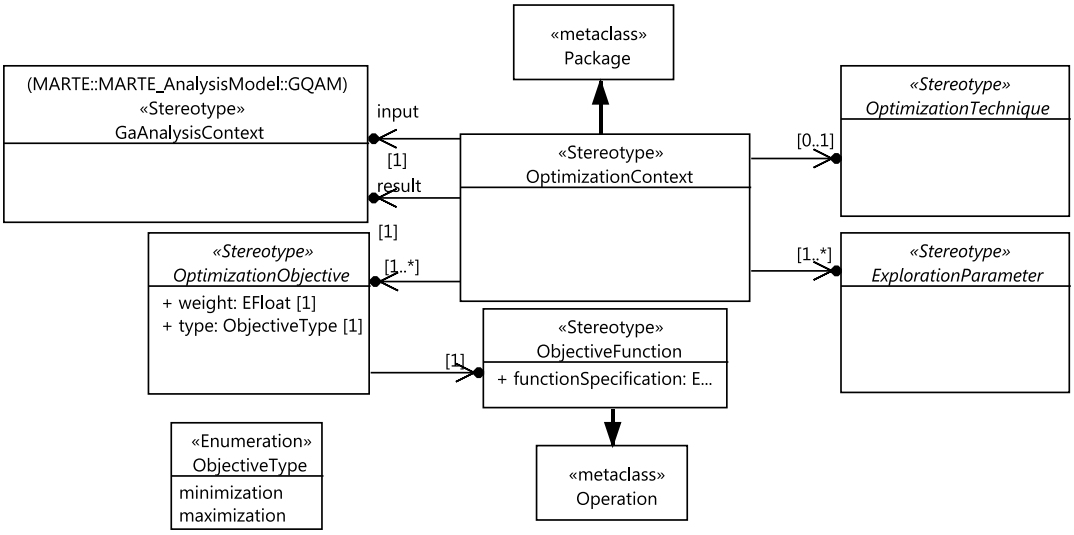

Figure 6. UML Profile pour le contexte d'optimisation

5) Les modèles d'optimisation ainsi que des modèles pour l'analyse et la spécification de l'architecture sont basés sur UML. L'utilisation de l'UML permet de faciliter l'intégration des différentes activités. Ceci est obtenu en majeure partie par l'ensemble des transformations qui automatisent des étapes importantes telles que la production du modèle AUTOSAR préliminaire à partir du modèle EAST-ADL2. En fait, la spécification d'architecture basée sur les concepts de EAST-ADL2 et AUTOSAR est réalisée dans ce travail par un mécanisme de profil UML. Le profil UML complet pour l'EST-ADL2 était disponible. Ce n'était pas le cas pour AUTOSAR et donc ce travail en définit un. Les modèles d'analyse sont établis avec SysML et MARTE pour lesquels les profils UML ont été définis et standardisés par l'OMG (Object Management Group). Les concepts pour l'optimisation ne peuvent pas être exprimés ni avec SysML, ni MARTE ainsi qu'EST-ADL2 et AUTOSAR. En conséquence, pour eux, un modèle de domaine est formalisé et son profil UML est défini comme susmentionné dans le cadre de la contribution 4.

Tous ces modèles, modèles d'architecture, d'analyse et d'optimisation avec des algorithmes d'analyse et d'optimisation peuvent être exécutés et ont été intégrés dans un cadre et structurés le long de couches d'abstraction et de points de vue. Le cadre lui-même (appelé AFfMAO – Architecture Framework for Modeling Analysis and Optimization) a été développé comme une instance d'un Cadre d'Architecture de

l'Automobile (ang. Automotive Architecture Framework - AAF) définie dans ce travail. AAF a été construit en suivant les principes du Cadre Architectural (ang. Architecture Framework - AF) définie dans la norme ISO 42010. Cette relation est représentée sur la partie gauche de la Figure 7. En substance, le cadre d'architecture est un ensemble de conventions, principes et pratiques pour la description des architectures dans un domaine et/ou communauté des parties prenantes. Par conséquent la spécification de l'AAF a été faite en définissant des points de vue de l'architecture avec leurs préoccupations, sortes de modèles et de règles de correspondance. Le côté droit de la Figure 7 présente la perspective détaillée de AFfMAO. Les informations pertinentes à partir de cette figure concernent les choix des techniques de modélisation, un ensemble de transformations, des algorithmes d'analyse et d'optimisation et plate-forme utilisée pour réaliser l'AAF comme AFfMAO.

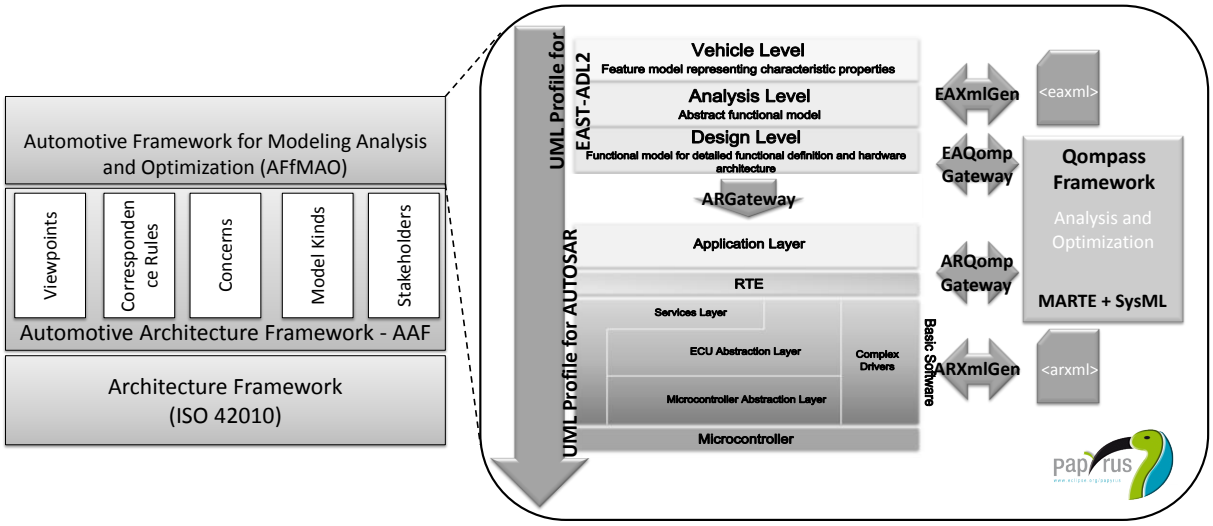

Figure 7. AFfMAO construit comme une instance de l'AAF

Les contributions de cette thèse apportent des solutions pour résoudre les problèmes cruciaux qui entravent la livraison d'un cadre pour une conception guidée des systèmes automobiles, alignés sur les principes de l'ingénierie dirigée par les modèles. Ils sont bénéfiques non seulement dans le contexte de ce cadre particulier, mais en général à ces constructeurs qui tentent de s'engager dans l'utilisation des standards EAST-ADL2 et AUTOSAR comme base de conception de leurs systèmes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

This introductory chapter gives an overview over the problems defining the scope of this work and lists the main contributions which aroused to handle them. For seek of clarity, it starts with a brief presentation of the context to which the thesis' problems relate. The context description will be broadened in the next chapter to provide to the reader an exhaustive synthesis of all the concepts fundamental to clear understanding of this work.

1.1. Context

Nowadays vehicle systems are marked by a wide range of software-based solutions that improve performance, safety and comfort of driving. Features like the self-parking system were beyond belief for ordinary drivers just 10 years ago. Not mentioning that we cannot still frame in our mind the vehicles driving autonomously, which is currently happening as few running examples were already prototyped.

From now on the term automotive/vehicle system will relate to those functionalities of a vehicle which are delivered through a combination of both software and hardware solutions. Automotive systems are perceived as *distributed*, *embedded*, *real-time systems*. First, software-based vehicle features are distributed on several embedded hardware components named Electronic Control Units (ECU) or on sensors/actuators. The application layer that spans over different ECUs is composed of the software components that can be delivered by multiple suppliers. The middleware is responsible for the communication between distributed software components. Each ECU runs an Operating System. All of this implies a distributed nature of the automotive systems (see Figure 1.1). Secondly their operation is tightened by the timing constraints of different kinds, e.g. the end-to-end response timing constraints. For instance the opening of the airbags during an accident should occur within 20ms. The last is a real-time constraint whose violation not only states the incorrect behavior of the system, but more importantly, can endanger a human's life.

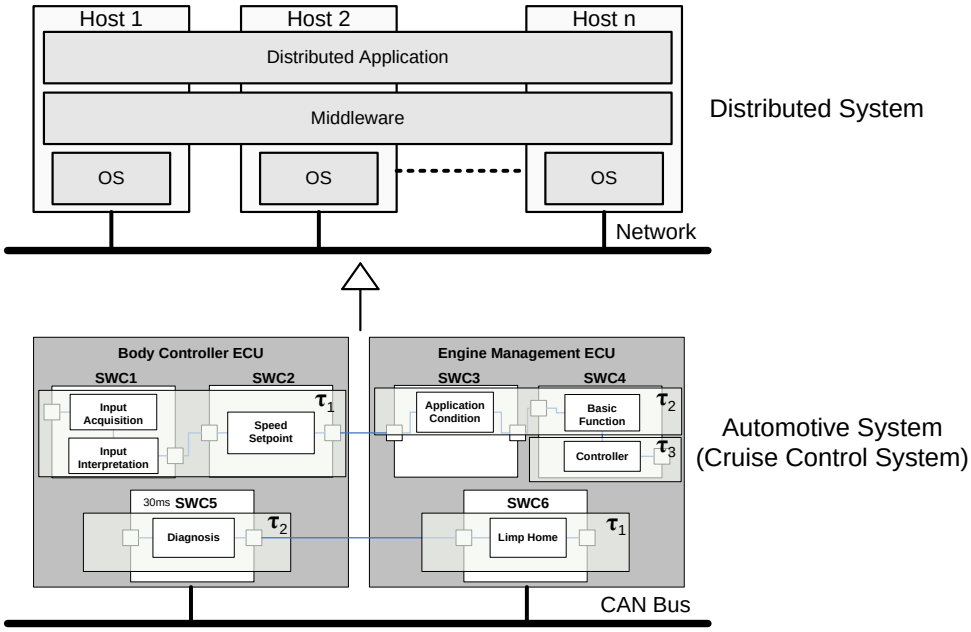

Figure 1.1. Distributed System and Automotive Distributed System

Automotive systems architectures (in short automotive architectures) are very complex high technology products. Architecture itself is defined as in the following Definition 1.1.

Definition 1.1 – Architecture: *fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution* [1].

Different factors contribute then to the complexity of automotive system architecture:

- **Size**: the number of features controlled by the software and hardware is substantial in nowadays premium cars. Within around 30 years the capacity of a code has increased from 0 to almost 10GB which implies millions of lines of code.
- **Distributed nature**: the first software-based solutions were very local and isolated. This was achieved by having only one feature per ECU. However the growing number of the SW features forced the automotive industry to shift towards distributed architectures. The "*one feature one ECU*" approach became very costly due to the increasing demand for the hardware elements. The additional motivation for that it was the shrinking physical space in the car which prevents further expansion of a hardware plant. As a result, nowadays automotive architectures are highly distributed, i.e. functions of the same feature span over different ECUs. The same ECU might host

functions of different features. This lead to a better optimization of the resources usage. Nonetheless it imposes a big challenge when designing such distributed architectures.

- **Real-time constraints**: correct functioning of a vehicle system is not only defined through the absence of functional errors but also by the means of a strict observance of the hard real-time constraints. Their existence serves primarily in a safety critical situations, like braking or during an accident when the airbags should be activated immediately. However in the competitive automotive world OEMs put high timing demands also on the features which don't have any impact on the safety aspect, e.g. features classified as the infotainment.
- **Safety requirements**: the safety aspect plays an important role as nowadays it is not only an internal concern of an OEM to provide reliable vehicles, but also a subject for regulations provided by the ISO [2].
- **Conflicting requirements**: all the different requirements like timing constraints, reduction of the hardware resources to lower the costs, provision of safety, etc. are in many cases orthogonal. This means that the satisfactory handling of one requirement can lead to the violation or deterioration of the others.
- **Sensitive to changes**: slight changes of a design or certain properties of particular architecture artifacts can lead to a radical modification of the architecture non-functional characteristics. For example, increase of an execution time of a single functional entity might lead to the violation of few timing constraints.

Due to this complexity that was and is still growing exponentially (as presumed to be for the next 20 years [3]), new strategies for design of the automotive systems need to be introduced. As a response, a number of initiatives have emerged, which either directly relates to the automotive systems or indirectly as they concern in general the distributed real-time architectures. Among them is the adoption of the model driven engineering (MDE) for the development of the automotive electronic systems [4]. The principle of the MDE approach is to incorporate abstract, in many cases graphical models to specify the functional and non-functional requirements, and finally, to produce a binary code that will fully respect the specification. The potential of the MDE has been spotted by the major car manufacturers and suppliers which initiated a project with a goal to provide a common standard drawing on the principles of the MDE. It is called

AUTOSAR (AUTomotive Open System ARchitecture) [5] and currently, this standard is the most influential in terms of modeling automotive systems. The development chain of the AUTOSAR methodology stretches from the depiction of application software components to the runtime infrastructure, including the description of the hardware platform. A related drawback of the AUTOSAR is its lack of support for the function-level modeling. Therefore there is a growing interest in combining this standard with the EAST-ADL2 [6] modeling language which targets the abstract, functional specification.

Besides the language aspect, the EAST-ADL2 and the AUTOSAR impose methodological rules for building the models. Their advantage is that they provide a common framework for the design of automotive electronic systems. However neither of those define how to perform certain design steps, e.g. how to distribute software components across hardware elements or how to map functional entities into OS (Operating System) tasks. In that respect, these two standards completely rely on a designer experience, increasing thereby the number of potential design flaws. As a consequence it is essential to conduct analysis like, timing or safety analysis to assure that the decisions made by the designer didn't lead to unfeasible architectures. We can go even beyond that and use techniques for Design Space Exploration (DSE) [7]. Their employment might not only assure the feasibility but in addition can optimize the key non-functional properties. This might lower the system final cost or increase the reusability of the architecture constituents.

1.2. Problem Statement & Motivation

As outlined in the previous subsection, clear and comprehensive view on an automotive system design, as well as its analysis/optimization, are crucial activities on a way to develop high-quality architectures. This requires appropriate modeling languages, analysis methods and techniques for enabling DSE. The general and initial objective of this thesis is to integrate these three activities within one complete methodological framework, supporting the design of the automotive architectures, followed by the EAST-ADL2/AUTOSAR methodology. Within this context, a set of problems aroused. As shown later, finding of appropriate solutions is significant to make possible the final integration of the above activities and provision of the guided seamless flow between them, to finally deliver optimized implementation model of an automotive system.

Having different types of models, i.e. architecture model, analyzable model and optimization model, is the main step towards the ability to perform an optimized synthesis of the software with

the hardware. The main phase of the synthesis is called deployment (see Definition 1.3). According to AUTOSAR, the deployment concerns the 1) **allocation** of the software components into the ECUs, 2) **partitioning** of the component's behavioral entities (so called runnable entities) in the OS tasks, and finally 3) **scheduling** of the OS tasks. The deployment step is done at the abstract modeling level which speeds up the design process as it is not delayed by awaiting the final code implementation. A crucial point for this step is about its validity in terms of its timing properties. Since the system refinement (of which deployment is an integral part) is done top-down, validity can be assured only under some assumptions abstracting lower-level details. A typical example is the assumption about the knowledge of the worst-case execution times (WCETs) of the AUTOSAR runnable entities. **It is obvious that the assumption about the precise knowledge of runnables' WCET before code implementation is most of the time unrealistic.** In many cases, implementation of certain runnables is re-used from previous systems, hence their WCET is known. However it is not true when new runnables are introduced. **This causes the problem in the attempt to deliver guided strategies for the architecture synthesis and in general, provision of an undisrupted top-down flow within the framework. What is even more significant is that the deployment itself in a way as it is defined by the AUTOSAR is not holistically supported by the existing techniques.** Although the amount of work that exists seems to be compelling and representative, there is a gap. **Proposed techniques either account for the OS tasks as the allocable entities or solve the problem in stages without consideration of a negative impact it has on a final results when compared to the holistic approach.**

Definition 1.2 – Synthesis: *derivation of a system from its specification.*

Definition 1.3 – Deployment: *synthesis step to determine the allocation of functional entities, their partitioning in OS tasks and assignment of priorities for the tasks.*

The EAST-ADL2 and the AUTOSAR specifications deliver a broad range of concepts that are necessary to define the complete system architecture. Recent efforts in a further extension of these standards provided even the capabilities to model the information needed for the timing analysis [8]. This triggered some work showing specific kinds of timing analysis (e.g. schedulability analysis) that can be run and how the timing model should be interpreted to do this [9]. The adequate work hasn't been done so far to handle the optimizations. **Though the field of**

real-time and distributed systems abounds in optimization techniques, there are no modeling concepts that would enable specifying an input necessary for this activity such as the optimization objectives (timing responses, memory consumption, etc.) or their priority, which plays an important role when optimizing orthogonal concerns. As a consequence the modeling and the analysis/optimization are not well integrated, namely consideration of these two problems simultaneously is not the case. This has resulted in many tools that deal either with the modeling or the analysis and/or optimization. As a consequence, it is difficult to embed these tools in the standard methodologies such as the EAST-ADL2/AUTOSAR methodology. This especially applies to the higher abstraction layers as those defined by the EAST-ADL2. It is a consequence of a high dependability of analysis/optimization tool on the platform specific information.

1.3. Contribution outlines

In order to enable the seamless development flow within the proposed framework this work offers a set of solutions to the aforementioned problems:

1) From the Design Space Exploration (DSE) techniques side, the main contributions relate to the definition of new techniques for optimizing the deployments. The proposed techniques are compliant with a way in which the deployment is defined by the AUTOSAR standard. That is to say, they consider the runnable entities as allocable units. Therefore the partitioning step is supported which is out of the scope of what current approaches are offering. Proposed techniques are based on evolutionary algorithms which is why they are able to handle large input architectures. This ability was evaluated by performing multiple tests, reaching 250 runnables. Accompanying significant aspect that is assessed this is the quality of the delivered, deployed architectures. This was done by comparing the results to those acquired with the exact methods or to the architectures for which the optimal deployment configuration was a priori known. Within the AUTOSAR, behavioral models might conform either to a *data driven* or *time driven* execution semantics, requiring hence to define different types of optimization strategies. The difference occurs on behalf of the timing analysis which diverse. Furthermore optimization metrics such as the timing or the memory are affected in an unlike way by the particular choices of a deployment. What also characterizes proposed techniques is the consideration of the multiple criteria (e.g. end-to-end timing responses, memory consumption) that defines a sound deployment configuration of the input architecture.

2) To improve the results of a deployment, this work suggests a refinement of the methodology uniting the EAST-ADL2 and the AUTOSAR methodologies. The purpose of doing so is to enable holistic consideration of the deployment problem, which as will be shown cannot be done with the current definition of this combined methodology. The impact of the changes is evaluated and shows significant improvement in regards to the metrics considered all along the deployment process.

3) In order to enable a qualitative deployment in regards to the optimization metrics, certain information is required. For instance to perform a deployment that optimizes the end-to-end response times, execution times of the runnable entities are necessary. As this information might be missing for certain runnables, definition of a new strategy for the architecture configuration is unavoidable. As a workaround, certain works propose to add to the methodology a special activity called time-budgeting. Instead of estimating worst-case execution times, the system integrator specifies so-called time budgets, i.e. constraints to the worst-case response times. Time budgets must be respected by the suppliers delivering the component implementation. The typical problem with this approach is that a supplier delivers the implementation of a particular component, which will be integrated as an interacting part of the system by the system integrator, in a later stage. Since the supplier will validate the component in isolation, without taking into account possible interferences of other components, it will be incapable of computing a correct worst-case response time (WCRT). In a sense if the component fulfills the time-budget constraint, the system integrator should take care of avoiding any possible interference with other components, which is not only a difficult task, but that typically inflates resource over-dimensioning. Such resource over-dimensioning turns into unsustainable costs for a mass-production. An alternative solution is the one in which time-budgets represent constraints to runnable's execution time, i.e. worst-case execution time (WCET) instead of WCRTs. The main question that remains now is how to specify this constraint. This thesis provides a solution, compliant with the AUTOSAR methodology and the one that will automatically assign the values for time budgets.

4) From the modeling side the first contribution is a specification of concepts essential to build an optimization model and to run Design Space Exploration techniques.

5) Optimization models as well as models for the analysis and architecture specification are based on the UML [10]. Usage of the UML serves to ease the integration of different activities.

This is attained for the most part by the set of transformations automating significant steps such as the generation of the preliminary AUTOSAR model out of the EAST-ADL2 model. In fact the architecture specification is achieved basing on the concepts of the EAST-ADL2 and the AUTOSAR and to model them this work uses a UML profile mechanism. The complete UML profile for the EAST-ADL2 already exists which is not the case for the AUTOSAR and hence this work defines one. The analysis models are established with the SysML [11] & MARTE [12] for which the UML profiles were defined and standardized within the OMG (Object Management Group). The concepts for the optimization cannot be expressed neither with the SysML nor MARTE as well as the EAST-ADL2 and the AUTOSAR. Consequently for them, a separate domain model is formalized and its related UML profile is defined.

Established contributions solve the crucial problems hindering the delivery of a framework for a guided design of the automotive systems aligned to the principles of the Model Driven Engineering. They are beneficial not just within the context of this particular framework, but in general to those OEMs who tries to engage the EAST-ADL2 and the AUTOSAR standards as the baselines for their systems.

1.4. Thesis Structure

Chapter 2 - Automotive Context: gives a general overview over the current trends in the automotive domain. It presents the main standards and the methodology for designing automotive architectures. The intent of this chapter is to provide detailed picture of the context and the fundamentals related to this thesis.

Chapter 3 – Challenges: lists and describes the main challenges identified for the automotive domain.

Chapter 4 – Approaches for the Computer-aided Configuration of the Automotive Architectures: relates to the contributions 1, 2 and 3 described in the section 1.3. It demonstrates a set of techniques contributing to the existing strategies used for the deployment of the real-time distributed architectures. It also presents strategies for the time budgeting and the refinement of the EAST-ADL2/AUTOSAR methodology. All this is evaluated to show the added value of the new techniques and the refined methodology.

Chapter 5 – UML based & Optimization-aware modeling of the Automotive Architectures: presents the contribution related to the UML modeling of the automotive architectures (contributions 4 and 5 described in the section 1.3). Above all it is the specification of the UML

profile that serves to express the optimization concerns. Accompanying is the presentation of the UML profile for the AUTOSAR and a set of transformations between different models (architecture model, analysis model and optimization model) that empower the overall integration of different activities within the framework.

Chapter 6 – Conclusion: concludes this dissertation and draws possible directions for the future work.

2. Automotive Context

This chapter starts by introducing the context of this thesis, i.e. automotive systems. Then it demonstrates the nowadays trends in the automotive domain by first discussing the employment of the Model Driven Engineering and then presenting the concept of an Architecture Framework The last establishes terminology based on which framework discussed within this work was built. Following is the overview of the most current and significant standards, i.e. the AUTOSAR [5] and EAST-ADL2 [6]. This chapter completes by presenting the methodology for designing automotive systems that is constituted by the EAST-ADL2 and the AUTOSAR.

2.1. Automotive System

Automotive systems these are distributed, embedded and real-time systems.

Embedded nature of the Automotive System: an automotive system is built of elementary subsystems where each can be classified as an embedded system. Elementary subsystem is made of so called ECU (Electronic Control Unit) of which central part is a microcontroller. This latter runs a set of functional entities (e.g. computation of a wheel torque) that are executed on it via a dedicated operating system. Functional entities are delivered as part of software components which define the interfaces to access them. Software component is a black box that hinders the implementation of contained functional entities. Often elementary subsystems interact with vehicle physical parts such as wheels or brakes through sensors/actuators. The software components of different ECUs in many cases have to communicate as they might contribute with their offered functionality to particular vehicle features (specific functionality offered within vehicle) such as the Cruise Control System. This leads us to the distributed nature of the automotive systems. Specific implementation of a vehicle feature is called subsystem where the last is built of elementary subsystems.

Distributed nature of the Automotive System: vehicle features incorporate large amount of functions thus it is not possible to run all of them on one microcontroller. Even if some of the features require less computational power, to decrease the overall cost of the system, and equally balance the load among the resources, functions will be distributed. Manner in which it will be done depends now mostly on a designer expertise. For these reasons, automotive systems have naturally evolved towards their distributed nature. As will be shown later when describing the automotive standards, the distributed middleware is among the standardized elements.

Real-time nature of the Automotive System: the correctness of certain vehicle features is evaluated against non-functional real-time constraints. The real-time constraints which implicitly affect the safety aspect are common in nowadays vehicles. For instance braking that is now controlled by software is a feature that should respect hard real-time constraints.

There is a large set of elements or concepts related to the automotive systems. Below are presented those which are the most common.

ECU

Electronic Control Unit is a computing unit that controls one or more subsystems in a vehicle. An ECU contains the hardware and software (firmware). The hardware incorporates electronic components distributed on a printed circuit board (PCB). The main component is a microcontroller. Among the others, there is a memory (EEPROM or flash memory), input/output interfaces to communicate with for instance sensors and a BUS communication unit, to send/receive messages on/from a communication BUS. The software (firmware) is stored either in the microcontroller or other chips on the PCB, typically in the EPROM or flash memory. Premium cars can have up to 80 ECUs. There exist multiple types of them depending on which mechanical part they control. These are electronic/engine control module (ECM), powertrain control module (PCM), transmission control module (TCM) and others, in total around 14 types.

Sensor/Actuator

Sensors are used to perceive the surrounding of a vehicle or the physical parameters of vehicle components like for example speed of a wheel. Actuators on the other hand affect the environment. They are controlling certain mechanisms in a car by introducing or preventing a motion. An example is an actuator for adjusting a vehicle idle speed.

Communication BUS

As highlighted before vehicle subsystems might be distributed over many ECUs. For most of the time, software functions of these subsystems need to communicate to fulfill the complex functionality. If communicating functions reside on different ECUs, these ECUs require a connection link. In the past, automotive OEMs were using point-to-point wiring systems. This turned out to be highly inefficient with an increase of the ECUs in a vehicle. To handle this problem, they replaced a dedicated wiring with in-vehicle networks. This reduced the overall

cost, complexity, and weight and enabled further expansion of the number of ECUs. There are different standards for the communication networks like CAN, FlexRay, TTCAN, LIN and Ethernet used for this purpose. The most popularized are the CAN and FlexRay. Recent efforts within the AUTOSAR group show the high interest in focusing only on the Ethernet. This topic will make one of the main contributions in the evolution of the AUTOSAR standard to the version 4.2. Below is a brief description of the mentioned standards for the bus.

- CAN (Controller Area Network) – this bus employs serial communication protocol, i.e. bits are sent one at a time, sequentially. It handles the detection of collisions, errors, retransmission of corrupted messages and the prioritization of sent and received messages. There are few versions of the CAN protocol where the most popular is so called High-Speed CAN. Its implementation involves two wires and allows communication at transfer rates up to 1Mbit/s. CAN is based on a broadcast communication mechanism. Each message sent on the CAN bus has identifier which is unique. It defines the content of a message and its priority. As multiple ECUs might try to send a message at the same time, CAN bus features a scheduler which operates on messages' fixed priorities. The scheduler is non-preemptive. Therefore before the transmission, arbitration process is run to decide whether a particular ECU can access a BUS. This will be possible only if the message that it wants to send, has the highest priority and currently, the bus is not processing any other message. This type of scheduling introduces non-determinism. Certain assumptions on the sending behavior of the nodes enable computation of worst case scenarios. This will be discussed more deeply in the section 4.4 when presenting techniques for the optimized synthesis.
- FlexRay – was developed by the FlexRay Consortium with the objective to have a bus with a higher bandwidth than the CAN and with the support for the time-triggered (TT) systems. Indeed, the data rate can reach up to 10Mbit/s. Secondly FlexRay bus can handle an event-triggered (ET) and a time-triggered systems. This is done by having two slots for the bus, so called static (for ET) and dynamic (for TT). Hence it offers the flexibility of the ET and timing guarantees and some fault tolerance which is characterizing the TT.
- TTCAN (Time-triggered CAN) – is an adaptation of the CAN bus to provide the capabilities of handling the time-triggered systems. The adaptation is done through the software in a higher layer, running on top of a CAN protocol.

- LIN (Local Interconnect Network) – was developed in order to create a low-cost communication standard. The CAN bus was too expensive to employ it for the realization of all the car features. LIN is a cheaper solution that can be used for less critical applications where the bandwidth and the versatility of the CAN are not required. It consists of a single master and one or more slave devices (up to 16). Communication is fully controlled by the master.
- Ethernet - is a widely known protocol for the Local Area Networks. Recently it is gaining more and more attention of the automotive players. The attractiveness of the Ethernet is due to the heavy and expensive wiring in nowadays cars. Therefore OEMs see the potential in the Ethernet as a technology that will reduce the complexity and cost of wiring. That is why the recent advances of the AUTOSAR standard are mostly focusing on the adoption of the Ethernet.

Gateway

These elements are used to enable the transfer of data between different sub-networks. In many cases networks communicated with the gateway are characterized by different protocols as one protocol cannot satisfy the requirements of all automotive applications. A typical gateway contains several interfaces corresponding to different networks such as CAN, LIN or FlexRay. ECU itself can serve as a gateway if it is connected to more than one communication bus.

Operating System

The ECUs, as these are embedded devices are controlled by the Operating System (OS). The high requirements on the timing behavior of the automotive systems require using a Real-Time Operating System (RTOS). The main feature of the RTOS is that it should serve the application requests in a real-time. In the automotive domain, OSEK/VDX [13] is the widely used standard for the RTOS. The AUTOSAR defines its own operating system called AUTOSAR OS [14] which is based on the OSEK v 2.2.3. The OSEK was created in 1993 by the consortium constituted by the main German players of the automotive market. In 1994 French car manufacturers which were leading a similar project called VDX (Vehicle Distributed eXecutive) joined the German consortium and hence the new name OSEK/VDX was established. Specification of the OSEK/VDX comprises three areas:

- **Communication:** exchange of a data within and between ECUs.

- **Operating System:** real-time execution of ECU software.
- **Network Management:** specification of protocols for managing the network.

Its initial version considered event-triggered systems. It was later extended with a support for time-triggered systems.

OS Task & BUS Frame

Operating System task is the execution entity considered by the scheduler. The last is responsible for the sequence of tasks execution. Task itself frames an execution of functional entities. BUS Frame has a similar meaning as an OS task. However it is related to the communication BUS and instead of running functional entities, it transfers the signals sent between them.

Scheduling

Scheduling is responsible for the assignment of hardware resources, i.e. either a processor or a bus to OS tasks or bus frames which represent the schedulable entities. Large set of algorithms for the scheduling exists, the choice of which impacts the execution sequence of the schedulable entities. Apart from the scheduling algorithm itself, there are other important factors affecting the final execution order, so called scheduling factors. These are the:

- hardware topology
- performance of the underlying hardware platform
- delays of the communication protocols
- allocation of the software entities to the hardware
- specification of the OS tasks (mapping of functional entities on them)
- specification of the bus frames (mapping of the messages on them)
- worst-case execution times of a functional entities or worst-case transmission times of a messages
- etc.

In the context of the embedded systems, there are two main scheduling strategies, the event-triggered and the time-triggered. These two are applicable for the scheduling of the communication and the processing of the OS tasks. They differ in the triggering mechanism for the start of communication and processing actions [15].

Event-triggered: assumes that all communication and processing activities are triggered when a significant event occurs. The event is unrelated with the clock tick but is realized by the interrupt mechanism [16].

Time-triggered: communication and processing activities are initiated by the progression of a real-time clock. The interrupt in this case is the real-time clock interrupt. The activities are triggered according to the predefined periodic pattern of clock ticks [17]. The time-triggered systems assume that the clocks of all the hardware nodes are synchronized. This introduces a notion of a global time that is available at every node. This is very strong assumption and not easy to maintain however indispensable in a distributed environment. This resulted in specifications of synchronization protocols which establish the common time among distributed processors. Time-Triggered Protocol (TTP) is an example of integrated communication protocol for time-triggered architectures [18]. It provides the services required for the implementation of fault-tolerant real-time systems such as predictable message transmission but also clock synchronization.

Each scheduling algorithm requires defining certain properties which are part of the scheduling factors. The process of their specification is also called scheduling. Therefore the term scheduling has two meanings. **1).** Scheduling as a process deciding on how to commit hardware resources – CPUs/BUSes between the tasks/frames. **2).** Scheduling as a specification of key parameters necessary to run a specific scheduler. Concerning the second definition the type of parameters to set is linked to the type of a scheduling used. If this is an event-triggered approach, then the main properties are the priorities of the OS tasks/bus frames. For time-triggered it is the specification of a schedule table which relates the clock ticks with the task/frames activation. The valuation of these properties has a direct impact on the scheduling algorithm and in consequence on the system behavior. The choices made can be evaluated. In the domain of the automotive systems this is an important step due to the multiple concerns, especially safety which is implicitly affected by the scheduler. Evaluation of the scheduling algorithms and of the values assignment done for crucial parameters determining the behavior of a scheduler, in the context of the real-time systems, is done against the real-time constraints. The analysis used to assess the schedule in regards to the timing constraints is called schedulability analysis. This work is focused on particular scheduling algorithms applicable for the automotive domain and for them existing schedulability analysis techniques will be presented. They are significant in the context

of the Model Driven Engineering as they enable to analyze abstract models for their correctness in terms of timing constraints.

2.2. Model Driven Engineering

The market of software applications grows rapidly as well as their complexity which can be expressed with their size, distribution, difficulty of the tasks to perform, etc. Automotive systems are great example of that. This constant evolution strives for the parallel amelioration of the approaches for building such systems. Efforts to do this moved us from the point in which the assembly code was a baseline for the development to the point in which object oriented programming is a common approach to proceed. Nowadays we can see a tendency of going even beyond that by employing abstract models as a starting point of a system specification. A model of a system serves as description or specification of that system but also of its environment.

Apart from the modeling Model Driven Engineering (MDE) encompasses the actual process of a system design using models and also models exploration. These two additional concerns are significant to fully and efficiently employ the modeling. First the definition of process or as it is called in many cases, methodology, clearly states the crucial steps to be performed in order to deliver the final system specification. Apart from the steps itself it is also appropriate sequencing of them. The main goal is to achieve at the end high efficiency of a development process and correctly order the activities to prevent delays. Clear methodology allows building a tool or combining a set of tools to support the development chain. The tools embed modeling of domain languages, models transformations, code generation or exploitation of the models. The last is essential to lead a qualitative design. The adoption of the MDE by the automotive OEMs is done even in a standardized form. The standard is called AUTOSAR (see subsection 2.4.1). Apart from the modeling artifacts, the AUTOSAR provides also the methodology of design and enriches the models with concepts, like timing information, that makes it possible to analyze them. The MDE for the automotive needs to also respond to the problems caused by the distributed nature of a system design. The business model employed by the automotive OEMs favors engagement of many suppliers delivering subcomponents of a system which are then integrated by the OEM. This has economic advantages but poses few challenges like appropriate exchange of information or architecture integration in this distributed environment. These problems are implicitly handled by the AUTOSAR. For instance definition of a common

language supports distributed environment. Others like timing information exchange are considered by the research, just like in [19].

Framework advertised in this work connects to the principles of the MDE. It employs the domain specific modeling languages as a way to define the architecture. Then it follows a predefined methodology and employs strategies enabling to exploit the models. Hence this thesis shows through its contributions, that there are still deficiencies in a current organization of the MDE approach as it is defined for the automotive domain.

2.3. Architecture Framework

The framework described in this work (see chapter 5) builds on the principles of an Architecture Framework (AF). Architecture framework is a set of conventions, principles and practices for the description of architectures within a specific domain and/or community of stakeholders [1]. It can be considered as a subset of the architecture description. This relation is shown on the Figure 2.1 where the right side of this figure depicts architecture framework and the left, architecture description highlighting the elements which belong also to the AF. As can be seen the AF consists of entities such as stakeholders, concerns, architecture viewpoints, model kinds and correspondence rules, as well as the relationships between these entities. The *stakeholders* of a system have *concerns*. Concerns evolve from requirements, design, or implementation choices. Stakeholders might be individuals, teams, or organizations. An *architecture viewpoint* establishes the conventions for the construction, interpretation and use of *architecture views*. The latter express the system architecture from the perspective of a specific set of concerns. An architecture view is part of the architecture description but not of architecture framework. This is because AF aims to define set of practices for building architecture description and architecture view is already a work product of architecture specification, following the conventions defined in AF. An architecture viewpoint contains at least one *model kind*, which defines the conventions for an *architecture model* (which is only part of an architecture description). Next, the *correspondence rules* govern the relations between the elements of an architecture description which are represented in the architecture description as *correspondences*. For example, a correspondence rule between a hardware platform and software components might require that each software component must be allocated to a particular hardware element. Finally the *architecture rationale* which is not part of an AF specification but of an architecture definition, records explanation, justification or reasoning about architecture

decisions that have been made, e.g. rationale for each architecture viewpoint included for use in particular architecture description. Notable examples of architecture frameworks of this type are DoDAF (Department of Defense Architecture Framework) [20] and MODAF (British Ministry of Defense Architecture Framework) [21].

Definition 2.1 – Architecture Description: *work product used to express an architecture [1].*

Definition 2.2 – Architecture Framework: *common set of principles and practices for creating, interpreting, analyzing, and using architectural descriptions for a given application domain or stakeholder community [1].*

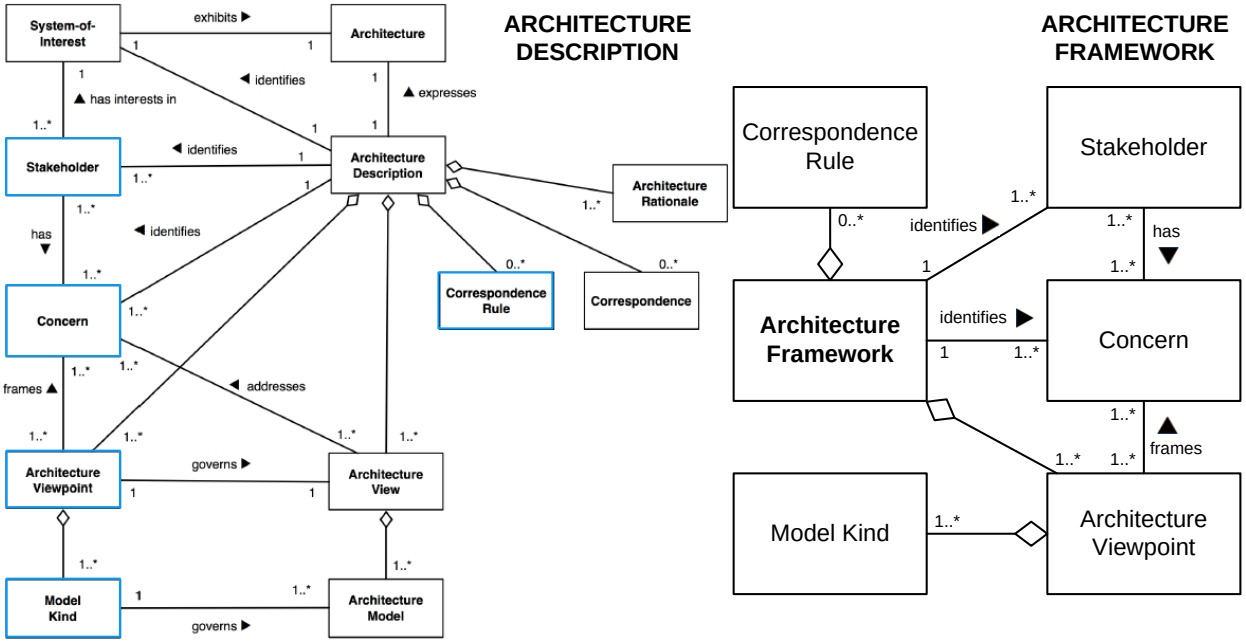

Figure 2.1. Conceptual Model of Architecture Description and Architecture Framework.

2.4. Automotive Standards

This subsection describes the AUTOSAR standard and the EAST-ADL2 modeling language.

2.4.1. AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is a joint initiative launched by BMW, Bosch, Continental, Daimler Chrysler, Volkswagen, and Siemens VDO in August 2002. The current version of AUTOSAR is 4.0. The main goal of this project was to create an open standard for automotive E/E (Electrics/Electronics) architectures, mainly to control their

complexity. AUTOSAR defines also a methodology for designing automotive systems and a way of describing their software architecture. System architectures developed based on the AUTOSAR consist of the layers shown in the Figure 2.2.

The Application layer contains a specification of software components which implement the desired functionality. It forms the basis for competition between OEMs.

The AUTOSAR runtime environment (RTE) provides communication services for application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuators). It enables AUTOSAR software components to be independent of specific ECU. RTE is an implementation of the VFB (Virtual Function Bus) which is an abstract communication environment. It does not specify which particular technology is to be used to exchange data. Therefore, the VFB enables AUTOSAR to be used on various communication platforms such as CAN or FlexRay. Definition of data exchanges between software components using the VFB enables them to be independent of the underlying hardware platform. Moreover, this allows concentrating directly on communications between software components without concerns as to whether data is transmitted within an ECU or between ECUs.

The BSW (Basic SoftWare) consists of many sub-layers. The highest layer of the BSW is the Services Layer. It provides the operating system, vehicle network communications, management services, memory services, and diagnostic services.

Next, the ECU Abstraction Layer covers I/O (Input/Output) and communication hardware abstraction, allowing higher software layers to be independent of the ECU hardware layout.

The Complex Drivers Layer bridges hardware and RTE. It provides non-AUTOSAR, special-purpose functionality, such as device drivers.

The Microcontroller Abstraction Layer is the lowest software layer of the BSW. It includes drivers with direct access to the microcontroller internal peripherals and memory mapped microcontroller external devices. Its purpose is to isolate higher software layers from the specifics of the microcontroller.

AUTOSAR itself is missing the ability to express functional aspects. It concentrates mostly on implementation issues, viewing the software architecture as the highest level of system abstraction.

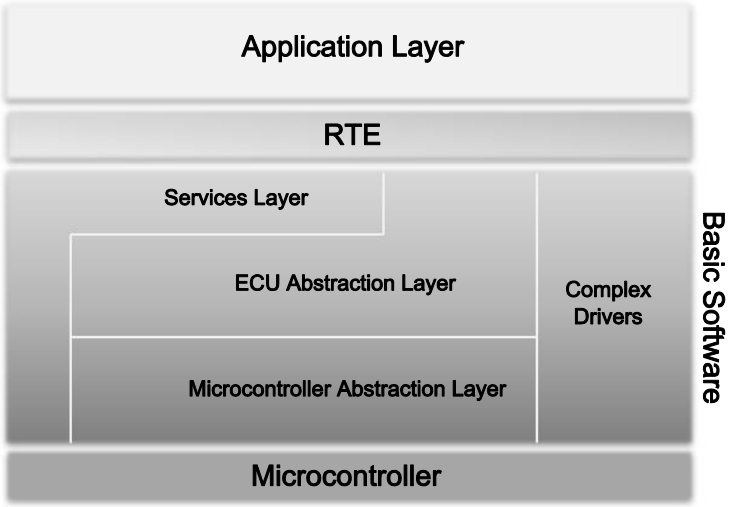

Figure 2.2. AUTOSAR Architecture Layers Schema

2.4.2. EAST-ADL2

The shortage in modeling constructs defining the functional characteristics of automotive systems was identified by the partners of the previously completed EAST-EEA project (Electronics Architecture and Software Technology – Embedded Electronic Architecture, see [22]) and the current ATESST I and II projects (Advancing traffic Efficiency and Safety through Software Technology) [23]. According to these findings, advanced and complex systems also require model-based design encompassing higher levels of abstraction and multiple concerns to support cost-efficient and effective development [24]. As a result, they defined the EAST-ADL (Electronics Architecture and Software Technology – Architecture Description Language) modeling language, refined subsequently in the ATESST project to EAST-ADL2 [6]. EAST-ADL2 is an architecture description language that provides modeling concepts for high-level architecture descriptions of automotive electronic systems. The AUTOSAR concepts are included as the implementation level of the EAST-ADL2. However a serious drawback of the EAST-ADL2 is its poor support for behavior modeling. This in turn prevents many kinds of early analyses of model validity.

Similarly as in the AUTOSAR, the EAST-ADL2 defines abstraction layers (see Figure 2.3). The Vehicle Level (VL) contains a feature model, i.e. specification of a vehicle features. The Analysis Level (AL) refines the VL, namely features are refined into functions which accomplish a work of features that they build. The mapping between features of the VL and functions of the

AL is m-to-n. This level makes no distinction between HW and SW. Design Level (DL) is the last level specified by the EAST-ADL2 concepts. At this level, functions from the higher level are refined into sub-functions and atomic (non-decomposable) functions. Also at this stage an abstract hardware platform is introduced. The Implementation and Operational Levels relate to the AUTOSAR. They are mentioned as part of the EAST-ADL2 to show that the purpose of this language is to complement the AUTOSAR.

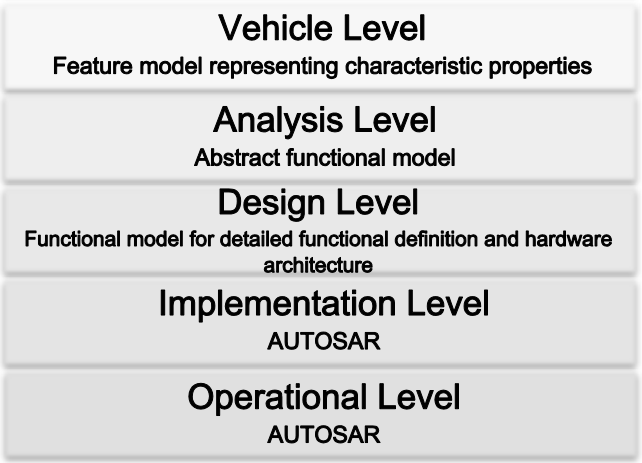

Figure 2.3. EAST-ADL2 Abstraction Layers

2.5. Methodology of Design (EAST-ADL2/AUTOSAR Methodology)

The EAST-ADL2 and the AUTOSAR next to the specification of domain concepts deliver also methodologies. These methodologies aim to provide a seamless and model-based development process and to give guidance on how to use the languages to construct system and software architectures.

EAST-ADL2 Methodology

This methodology doesn't intend to impose a fixed software development process [25]. It is a consequence of a large number of activities of the EAST-ADL2 for which companies have already developed certain approaches to proceed. In principle it divides the activities into two sets; *Kernel Methodology* and *Extensions*. The *Kernel* represents the essential development activities comprising a top-down, central, constructive phases, necessary to produce complete

architecture model. These activities are grouped accordingly to the layers of the EAST-ADL2 and traverse them in a top-down way. They are as follows:

- Vehicle Modeling – activity done at the Vehicle Level consisting in modeling vehicle features and in specifying related requirements.
- Analysis – activity done at the Analysis Level to create a functional analysis model.
- Design – activity done at the Design Level to create a functional design model. It employs a specification of a software and hardware entities separately. Most importantly this stage of the methodology requires mapping software entities represented by functions on the hardware elements.
- Implementation – concerns the implementation of a hardware and software and configuration of a final solution. The models developed at this stage these are platform specific models where platform is described by the AUTOSAR concepts. Configuration at this level, according to the AUTOSAR is characterized by steps presented in more details below, in the context of the AUTOSAR methodology. Please note that the AUTOSAR is not mandatory to be used at this stage.

The *Extensions* part talks about activities related to the modeling of environment, variability, behavior, etc. It also refers to the analysis of such non-functional concerns as timing or safety assurance.

AUTOSAR Methodology

The development chain of the AUTOSAR methodology [26] (see Figure 2.4) stretches from the depiction of application software components to the runtime infrastructure, including the description of the hardware platform. This methodology chain is specified through the following phases:

- **Vehicle Architecture Design:** During this phase, the application is specified in terms of the *software architecture*: software components, interfaces, ports and connectors. The platform is specified in terms of *hardware architecture*: ECUs and their interconnection topology, i.e. physical ECUs interconnection through buses or dedicated links. The mapping of software components on ECUs is not done during this phase, but constraints on this mapping can be specified at this level. The vehicle architecture design models are

exchanged through an XML artifact called *System Configuration Input*, which actually serves as input for the following phase.

- **System Configuration:** During this phase the mapping of the software architecture into the hardware architecture is performed. Software components are mapped into ECUs, and application messages are mapped into bus frames. Moreover, the internal behavior of software components is also specified. Internal behavior is a specification of events (RTEEvents) and runnable entities. The latter are the smallest code-fragments provided by software components. The artifact to be produced at the end of this phase is called *System Configuration Description*, which serves as input for the following phase.
- **ECU specific information extraction:** During this phase information specific to each ECU is automatically extracted, and a first layer of RTE is automatically generated. The artifact to be produced at the end of this phase is called *Extract of System Configuration Description*, which serves as input for the following phase.
- **ECU configuration:** During this phase the basic services of the platform are configured on each ECU. The most important step lies in the specification of the mapping of runnable entities into OS tasks. The artifact to be produced at the end of this phase is the *ECU Configuration Description*. This artifact is used for the generation of binary code.

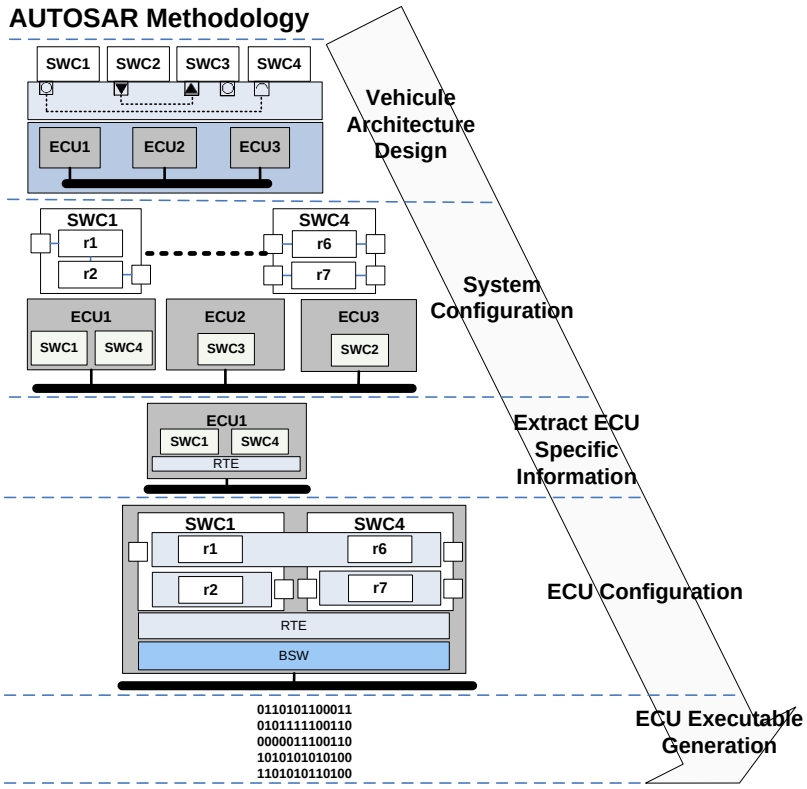

Figure 2.4. AUTOSAR Methodology

Combined EAST-ADL2/AUTOSAR Methodology

The intent of the EAST-ADL2 is to complement the AUTOSAR with a functional level modeling. Consequently, specification of the EAST-ADL2 layers contains a direct reference to the AUTOSAR which defines the Implementation and Operational Level of the EAST-ADL2 language. Similar reference to the AUTOSAR is done when specifying the EAST-ADL2 methodology. This shows a desire to reason about the overall development process as a combination of the activities done at the EAST-ADL2 level and then, at the AUTOSAR level. This concept hasn't yet found its reflection in current practices of the automotive OEMs. It is due to the fact that the EAST-ADL2 itself hasn't been yet fully adopted by the car manufacturers. Still low but constantly increasing tools support and a lack of language stability are the major issues preventing to recognize the EAST-ADL2 as an automotive standard. Also crucial are the gaps of the EAST-ADL2 with respect to company specific processes and needs and simply lack of engineers familiar with this relatively new language. However ongoing efforts to improve the EAST-ADL2 within projects such as MAENAD (Model-based Analysis & Engineering of Novel

Architectures for Dependable Electric Vehicles) prove the growing interest in using its constructs. Therefore the framework defined within this work accounts for the EAST-ADL2 which implies employment of the EAST-ADL2/AUTOSAR methodology.

The combination of the EAST-ADL2 and the AUTOSAR methodologies is not straightforward because certain decisions, done at the EAST-ADL2 level influence the activities of the AUTOSAR level. An important example would be an allocation of functional entities into the hardware elements, which is done at the design level of the EAST-ADL2 methodology. This determines the allocation of the runnable entities which normally would be done through the allocation of software components at the AUTOSAR level. Fixing the allocation on the EAST-ADL2 level means that the allocation on the AUTOSAR level can be deducted due to the correspondences between EAST-ADL2 functions and AUTOSAR runnable entities. Additional challenge is a switch between the EAST-ADL2 and the AUTOSAR itself. These two languages operate on different concepts. Hence it is obligatory to define the mapping between them. This is presented later in this work in the subsection 5.5.1. The transformation requires additional activity in the methodology to specify how the functional entities will be transformed to runnable entities and how these generated runnables will be then embedded in the software components. Therefore the final, EAST-ADL2/AUTOSAR methodology would be that of presented on the Figure 2.5. According to it the activities of the EAST-ADL2 level remain unchanged. Next is the activity *Generation Specification* done to decide on the generation of the runnable entities and their grouping within the software components. Product of this activity called *Generation Model* enables to compose software architecture. The way in which runnables are grouped might be decided based on the distribution of the runnables implementation task among the suppliers. Suppliers deliver entire software component hence runnables implemented by different suppliers cannot be put in the context of the same software component. If the transformation model is not provided, generators might employ predefined strategies to handle such cases. For instance they might assume one-to-one mapping between atomic functions from the EAST-ADL2 level and runnable entities of the implementation level. Software components can be generated in one of the following ways: one sw component for one runnable entity, one sw component for all the runnables allocated on the same ECU or finally they can simply reflect the compositional specification done at the EAST-ADL2 level [27]. Now it is possible to directly produce the System Configuration Description. It is important to note that the last needs to also include the

specification of the SW components allocation. This can be inferred from the allocation of functional entities modeled at the EAST-ADL2 level and the transformation model. The activity *Transform EAST-ADL2 to AUTOSAR* which outputs the System Configuration Description can now be done either manually or automatically. Automatic generation is used within the framework proposed in this work and detailed in the subsection 5.5.1.

The rest of the activities and their final products remain unchanged in regards to the original AUTOSAR methodology.

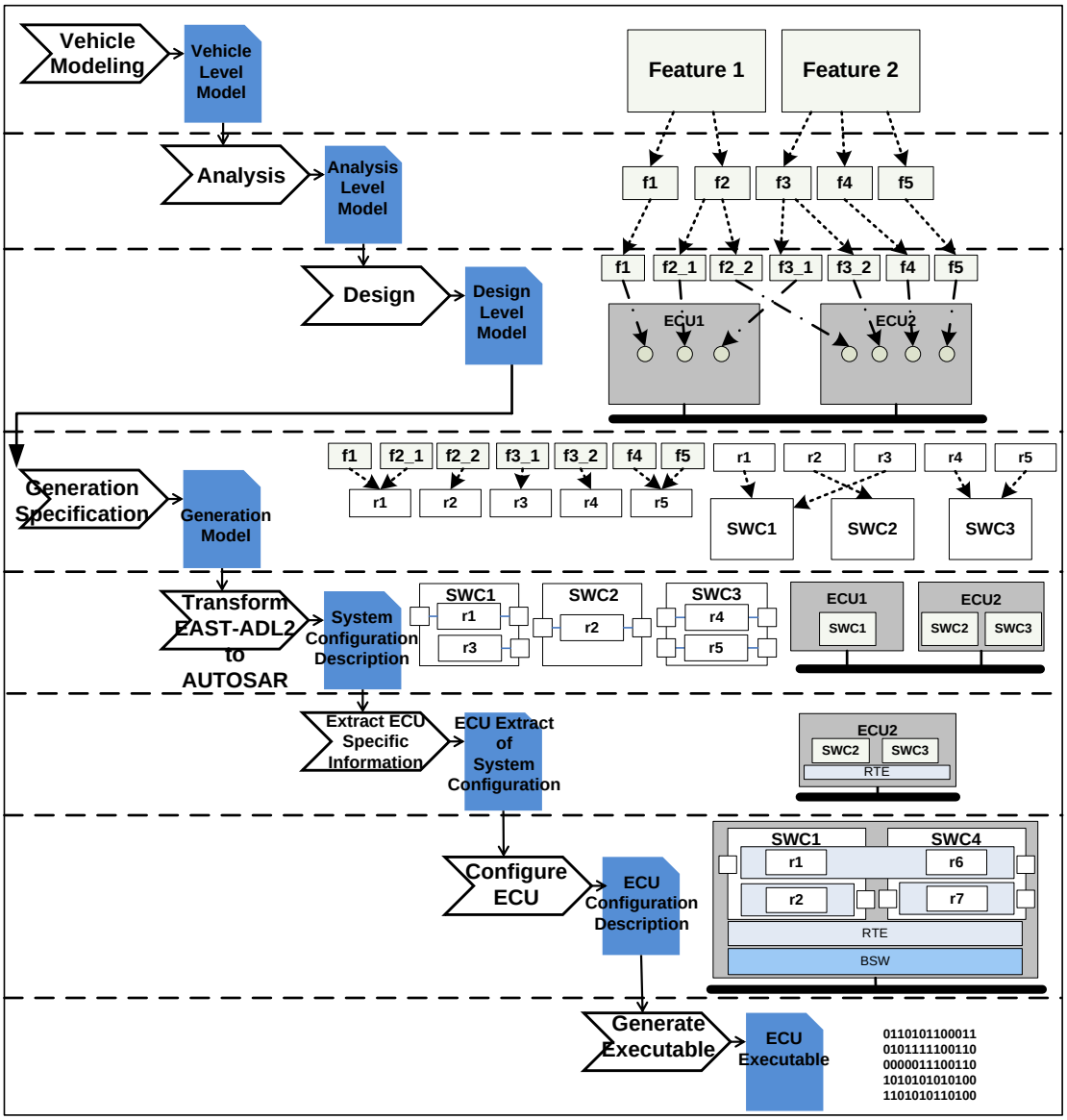

Figure 2.5. EAST-ADL2/AUTOSAR Methodology

3. Challenges

The diversity and scope of activities to be done when defining automotive architecture is substantial hence automotive domain is characterized by a vast range of challenges. This chapter highlights two main challenges which draw a borderline for the contributions of this work (section 3.2 and 3.3). Their choice is justified by presenting their significance in advancement towards qualitative development process. Compelling related work that confirms the fact that they still didn't find appropriate handling will be cited and discussed when presenting particular contributions in the next two chapters. Their description follows the section 3.1 which briefly highlights other relevant concerns of the automotive domain.

3.1. General Challenges

As stated in section 1.1 different characteristics of the automotive systems contribute to their complexity. Among the many are their size, distributed nature, real-time constraints, safety requirements, conflicting requirements and sensitivity to changes. This complexity is expected to grow exponentially as software opens wide range of possibilities to design more efficient and safer vehicles. A workaround is to focus on finding new approaches for an adept design. This poses new challenges such as employment of model based design, handling of a distributed design process, establishment of standards, consideration of safety related issues in a design process or finally education to teach new culture of systems design.

Challenges for Model Based Design

These include new ways of architecture description such as the definition of abstraction layers for architecture specification. In order to cover these abstraction layers one of the challenges is to define domain specific modeling languages. They should not only provide the means to model elements of architecture but be sufficiently expressive to enable architecture analysis and qualitative exploration.

Challenges in Distributed Design Process

A key issue is a software engineering process which involves many first and second tier suppliers. This so called distributed development environment strives for new ways of management during the system design. First, distribution of tasks needs to be well coordinated, especially as there are multiply dependencies between output and input artifacts coming from different suppliers. Incorrect handling of this might cause significant delays in the delivery of the

final product. Quite notable problem is a coherency in understanding and interpretation of particular concepts or modeling constructs. Usage of common standards or well-known modeling languages such as UML or SysML can overcome this issue. Unfortunately these languages in certain cases are not expressive enough to model all the concerns related to the automotive domain.

Challenges for Standardization

It should be of a high interest to standardize these parts of the architecture that play crucial role during the synthesis process. Namely the overall architecture is built of subsystems which might be delivered by different suppliers. In fact subsystems itself might contain components coming from different vendors. Therefore to enable the communication between these components or subsystems it is essential to establish common communication protocols or description of interfaces so the synthesis will be possible. The establishment of the AUTOSAR standard and also the EAST-ADL2 language responds to that need. However still, incoherent interpretation of them leads in some cases to incompatibilities between different tools.

Challenges for Safety

Automotive OEMs needs to assure that their system will meet stringent constraints on fault tolerance and reliability. Nowadays it is not only their internal requirement driven by a desire to stay competitive on the market. It is prevailed through the international regulations to comply with safety concerns as expressed in the safety standard ISO 26262 [28]. This necessitates elaborating on the ways that will prove the conformance to this standard, so the vehicle can be considered as safe.

Challenges in Education

Lastly a big challenge is a switch in mentality of the software engineers' community developing automotive systems. Engineers look at the system from the low level perspective such as an implementation code instead of a system level. If the advancements in the MDE for automotive are to be fully exploited, it is essential to acquaint new generation of engineers with a theoretical and practical knowledge related to the model based approaches.

Since 2006 when Broy et al. [3] listed main challenges faced by the automotive domain significant advancements have been made. This is reflected in the numerous tools offered on the market, steady progress of the automotive standards (e.g. AUTOSAR or ISO 26262) and a research that is highly interested in the challenges posed by this industry. However there are still

significant deficiencies that remain unresolved. From among them this work addresses two, related to techniques for the optimized configuration of automotive architectures and architecture description specification.

3.2. Configuration of Automotive Architectures

Future engineering aims in optimizing not a single component of a system but the entire architecture from a system level perspective. This holistic approach leads to better optimization results as it considers the dependencies between the system components which have high influence on the final system non-functional properties. If a development follows a top-down approach, in an optimal scenario, synthesis of architecture from its components should be done at the abstract models level. Optimal in this case refers to the overall speedup of the design. When configuring architecture without awaiting the final implementation of system components, as soon as the implementation is done, system can be integrated and run. Even abstract model when synthetized can serve for further architecture evaluation using simulation or static analysis techniques. This gain of time might not be noticeable in some domains, but is significant for the automotive systems design due to the distributed nature of this process. Of course lack of implementation implies deficiency of information that in many cases is necessary to lead a qualitative synthesis. Source code is essential to estimate the execution times of functional entities either using a static methods or simulation techniques. **Execution times or in fact worst case execution times (WCET) are requisite to run schedulability analysis test which enables to assess feasibility of architecture but is also used throughout the search of an optimal in terms of end-to-end responses configuration.** Their lack and desire to configure architecture at the early stage crave for new approaches that will still allow leading qualitative synthesis. As a workaround, automotive methodologies like in [29] propose to add a special activity in which so-called time-budgets are specified. Time-budgets are specified on software components, establishing deadlines for the functional entities the component encapsulates. These deadlines represent **1)** the constraints that have to be respected by the suppliers delivering the component's implementation and **2)** the execution times used by the system integrator to configure the software architecture. **So far there is no satisfactory approach for time budgets assignment mainly due to the NP-hard nature of this problem and a lack of criteria to drive their specification.** There is also a ubiquitous perception of what time budget should represent, whether constraint imposed on the worst case response or worst case execution time. If it restricts

WCRT their specification doesn't account for the concurrency which results from the hardware resources contention due to the other components allocated on the same ECU. This implies that a supplier delivering a component cannot validate in isolation (out of the overall system context) if the implementation he will hand over, respects the time budget. Budgeting WCET overcomes this issue but on the other hand poses a bigger challenge.

The related works as will be explained in more details later are characterized by certain deficiencies. First, most of them budget worst case response times and secondly they assume as an input the deployment specification. The latter simplification has a huge disadvantage. The deployment couldn't have been done in a qualitative way as time values were missing. Therefore this work will advertise a technique incorporating an idea of interleaving the process of time budgets specification and deployment. For this technique to run an adequate approach to deploy software architecture into hardware is required.

A considerable body of work targets the problem of deployment of distributed architectures and their optimization. Deployment step has a huge influence on non-functional properties of an automotive architecture. It impacts multiple crosscutting concerns such as the load balancing among ECUs or BUSes, end-to-end response times, memory consumption or safety. Number of possible deployment configurations increases exponentially with the number of hardware elements, functional entities or signals exchanged between them. Hence the space to explore is huge which means that the manual approaches as well as the extensive algorithms cannot be efficient. The deployment problem is not new and lots of techniques were already proposed. Nevertheless as will be shown later, none of the approaches is applicable for holistic handling of a deployment defined as allocation of functional entities, their partitioning in OS tasks and priorities assignment. Related works either consider OS tasks as allocable entities and hence the partitioning is out of their scope or treat the problem in stages without consideration of how the staged approaches might affect the final result in comparison to the holistic approach. **On the canvas of this shortage set of contributions aroused. This is elaboration of techniques based on the evolutionary algorithms which are responsible for the deployment.** In addition this work deliberates on their scalability and presents a technique to improve it. The scalability issue is not well addressed in other works although it is highly relevant especially in the automotive domain.

3.3. Architecture Description Specification

Modeling languages for expressing main concepts of the automotive architectures are reaching a high maturity level. They also tend to be enriched with a constructs to model non-functional properties and constraints. Hence the overall architecture description is composed of multiple models that relate to different concerns, spanning across abstraction layers. This leads to the first problem which is an overall integration. According to Broy et al. [4] an attractive vision is an integrated modeling approach that captures the relationship between all the models, and where parts of some models are generated from the models used in previous editions. Integration in this case boils down to the models refinement and mapping between the concepts of different modeling languages. The last requires defining a rigid mapping between the modeling constructs.

This work extends also the integration to transitions between activities such as modeling, analysis and optimization. Concerning the integration of the activities a current problem is that there are numerous complementary tools that support only certain activities from the all possible. Separate tools serve for the modeling and synthesis, e.g. SystemDesk from dSpace [30] or DaVinci Developer from Vector [31]. Other set of tools can be used uniquely for the analysis (SymTA/S from SymtaVision [32] to run timing analysis) or just optimization during the architecture synthesis (SynDEx [33]). As for the modeling, there is a wide range of tools adapted to specific needs of the automotive domain. It is not the case for the analysis or optimization tools. In fact there exists no commercial tool support for optimized synthesis of automotive architectures. This observation demonstrates the direction of the evolution of the MDE in a commercial context. Namely, first need was to provide languages and tools supporting them. Next is functionality for analysis which starts to evolve. Then we should expect the emergence of the tooling for optimization or optimized synthesis. Finally it would be of a high value to integrate all these tools or even more, embed all the activities and their related models within one tool. Nevertheless for the moment tools in many cases use their own modeling constructs or specific interfaces to connect to them which makes the overall integration hard to accomplish.

This work is highly concerned about the integration problem. As shown it is an interesting and vivid challenge of which proper handling will contribute to the speedup of the design process. Integration is achieved within the boundaries of the proposed framework, defined as an instance of an architecture framework (AF). The last implicitly serves for the integration. Each framework defined as an AF needs to specify correspondence rules between the models.

These can serve as guidelines for the designers on how to transform between models or for the tool suppliers to deliver automated transformations.

The goal is not to only integrate the models but also the three activities, i.e. modeling, analysis and optimization. To do this a definition of an analysis and optimization models is substantial. The EAST-ADL2 and AUTOSAR deliver the concepts that can be used for triggering analysis of timing properties. **The second concern can be modeled with neither of these two languages. This poses an interesting challenge for integrating optimization in a model based design.**

3.4. Conclusions

This chapter presented essential challenges that drive the current industrial and research activities of automotive domain. Part of the challenges aroused due to the desire for employment of model based design which requires definition of languages and implementation of tools support. Secondly the great challenge this is safety which currently plays the dominant role in the further evolvement of the AUTOSAR standard. There are many other challenges from which those that refer to the configuration of automotive architectures and their modeling analysis and optimization state the main focus of this work. The next two chapters provide a set of solutions that further advance the current state of practices related to the challenges of interest. Chapter 4 describes techniques for configuration of automotive architectures, in particular deployment and time budgets specification. Chapter 5 presents framework encompassing the modeling, analysis and optimization of automotive architectures by presenting modeling constructs that enable to integrate these activities.

4. Approaches for the Computer-aided Configuration of the Automotive Architectures

This chapter is focused on contributions related to the computer-aided design of the automotive architectures. These are the techniques used for an optimized deployment of architectures and time budgets assignment. The beginning of this chapter, section 4.1 formalizes common notions used throughout this chapter. Its two subsections formalize two activation models, i.e. data driven and time driven correspondingly in the 4.1.1 and 4.1.2. They also introduce additional concepts which are specific for each of the activation models. Schedulability analysis test is described in the section 4.2. This test is used to assess the feasibility of the architecture in regards to the designated deadlines for the end-to-end flows. It is also used during the search of the optimal deployment configuration. Different activation models require different schedulability analysis tests. The following is section 4.3 which discusses a refinement of the EAST-ADL2/AUTOSAR methodology. The main intention of the refinement is to adapt the methodology to the proposed configuration techniques. This has a very pragmatic reason as the change allows handling the deployment problem holistically which ultimately leads to the amelioration of optimization metrics. This gain will be evaluated in the section 4.6, after the specification of the deployment techniques in sections 4.4 and 4.5. Lastly, section 4.7 is devoted to the time budgeting problem.

4.1. Formalism

As already discussed, the deployment is done at the implementation level whose specification follows the AUTOSAR. Therefore certain notions come directly from this standard. Accordingly the input system model consists of two graphs. The first one is the AUTOSAR execution model represented by a directed graph $G_e = \{V_e, E_e\}$ in which V_e is the set of vertices representing runnables and E_e is the set of edges related to the links between them. Links model communication between the runnables and implicitly their precedence relation by specifying the source and receiver of the data. The second one is an undirected graph $G_h = \{V_h, E_h\}$ that expresses hardware architecture. Nodes represent hardware resources and the edges represent communication links between them. The hardware resources are ECUs and communication

buses. The remaining notions used throughout this work were gathered in the table below for a better readability.

Concept	Definition
E	Set of ECUs
ecu_i	ECU
β	Set of BUSes
b_i	BUS
$\Lambda(b_i)$	This function returns a set of the ECUs communicating through the bus b_i
R	Set of runnable entities.
r_i	Runnable entity
P_{r_i}	Period of a runnable entity
R_{r_i}	Response time of a runnable r_i
$E(r_i)$	ECU on which the runnable r_i is allocated
$\vec{C}_{r_i} = (C_{r_i,e_1}, C_{r_i,e_2}, \dots, C_{r_i,e_n})$	Worst case execution time of a runnable, characterized by a vector of WCETs, due to the heterogeneity of the hardware nodes. Later in this work the second index specifying the ECU is omitted, just for the simplicity of the notation.
swc_i	Atomic software component. Its behavior is defined by the runnable entities.
$SC(r_i)$	This function returns atomic software component of a runnable r_i.
PS_{r_i}	Communication ports of runnable r_i
$P_{r_i}^{in}$	Set of input ports of the runnable entity r_i
$p_{r_i,j}^{in}$	j^{th} input port of the runnable r_i
$P_{r_i}^{out}$	Set of output ports of the runnable entity r_i
$p_{r_i,j}^{out}$	j^{th} output port of the runnable r_i
$£$	Set of links
l_i	Link – represents an interaction between runnables
S	Set of data signals

s_i	Data signal – runnables exchange data through the signals on the links from the £
$Snd(s_i)$	Function returning runnable entity that sends a data signal s_i
$Rcv(s_i)$	Function returning set of runnable entities receiving a data signal s_i
P_{s_i}	Period of the signal s_i. It is equal to the period of a writer runnable.
$B(s_i)$	BUS on which signal s_i is allocated.
$\vec{C}_{s_i} = (C_{s_i,b_1}, C_{s_i,b_2}, \dots, C_{s_i,b_n})$	WCTT (Worst Case Transmission Time) of the signal s_i when transmitted on the BUS. It is assumed that the intra-ECU communication takes zero time. Also here for the sake of simplicity, the second index indicating the bus is omitted when relating to the signal WCTT.
$DS(s_i)$	Size of the data signal s_i
T	Set of the OS tasks
τ_i	OS task – the code of a runnable entity executes in a context of an OS task
P_{τ_i}	Period of the task τ_i
π_{τ_i}	Priority of the task τ_i. Fixed priority systems are the focus of this work.
$E(\tau_i)$	ECU on which the task τ_i is allocated
C_{τ_i}	WCET of the task τ_i. It equals to the sum of WCETs of all the runnables partitioned within this task, i.e. $C_{\tau_i} = \sum_{\tau(r_j)=\tau_i} C_{r_i}$.
$\tau(r_i)$	Task in which r_i is partitioned
π_{r_i}	Priority of the runnable entity r_i. It equals to the priority of $\tau(r_i)$, i.e. $\pi_{\tau(r_i)}$
M	Set of messages
m_i	BUS message
π_{m_i}	Priority of the message m_i
$m(s_i)$	Message that transmits the signal s_i
π_{s_i}	Priority of the signal s_i. It is equal to the priority of $m(s_i)$, i.e. $\pi_{m(s_i)}$
C_{m_i}	WCTT of the message m_i. It is a sum of WCTT of all the

	signals partitioned in this message, i.e. $C_{m_i} = \sum_{m(s_j)=m_i} C_{s_i}$.

Table 4.1. Basic Architecture Elements

4.1.1. Data Driven Activation

In order to describe this model, few additional concepts need to be added. These are defined below in the Table 4.2.

Concept	Definition
$\xi = \{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}$	Set of transactions
Γ_i	Transaction – each transaction is a 2-tuple, $\{R, £\}$. This work considers linear transactions.
$\Gamma(r_i)$	This function returns a transaction to which runnable r_i belongs
e_i	External event of a transaction Γ_i. Each transaction is triggered by an event which can be sporadic or periodic.
P_i	Activation period or an inter-arrival time of an event e_i. Implicitly it is the period of a transaction Γ_i. Runnables and signals within a transaction inherits their period (respectively P_{r_i} and P_{s_i}).
D_i	Deadline of a transaction Γ_i
R_i	Response time of a transaction Γ_i

Table 4.2. Additional Concepts for the Data Driven Activation Model

In the data driven activation model each transaction Γ_i is triggered by an external event e_i. Subsequent runnables are activated upon the completion of the predecessor runnable and retrieval of a data signal sent by the predecessor (if local) or the arrival of the message delivering the data values for its incoming signal (if remote).

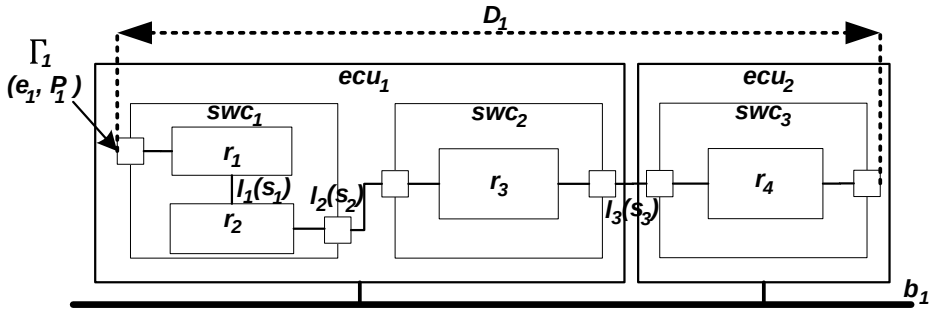

Figure 4.1. Data Driven Activation Model

4.1.2. Time Driven Activation

The time driven activation paradigm requires defining some additional concepts. They are presented in the Table 4.3.

Concept	Definition
$\xi = \{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}$	Set of paths[1]
Γ_i	Path - each path is defined as an ordered interleaving sequence of runnables and signals defined as $\Gamma_i = [r_{o_1}, s_{o_1}, r_{o_2}, s_{o_2}, \dots, s_{o_{k-1}}, r_{o_k}]$. $src(\Gamma_i) = r_{o_1}$ is the path's source and $snk(\Gamma_i) = r_{o_k}$ is the sink. Multiple paths may exist between each pair of source-sink.
D_i	Deadline of the path Γ_i
R_i	Response time of the path Γ_i
e_{r_i}	Event of a runnable entity r_i
Ω	Set of shared resources
σ_i	Shared resource. Runnables communicate by sharing data signals accessed through their ports. Data signal can be communicated either through a shared resource or via a message passing. The identification of shared resources uniquely depends on the runnables allocation and partitioning. For each identified data signal communicated between runnables of different tasks but of the same ECU a shared resource is defined. Data signals

[1] Please note that a path and a set of paths use the same representation as transaction and a set of transactions. This is to simplify the overall representation by minimizing the amount of used notation and also due to the semantic proximity of these two concepts, namely transaction and path.

	communicated between runnables of the same task don't require defining a shared resource. Also, for the inter-ECU communication no shared resource is required as in this case data signal is communicated by the BUS message.
$e(\sigma_i)$	ECU on which shared resource σ_i is specified
$r^w(\sigma_i)$	Set of writer runnables, writing to the shared resource σ_i. This work considers one-to-many communication hence $r^w(\sigma_i)$ contains only one writer.
$r^r(\sigma_i)$	Set of readers of the shared resource σ_i
$\zeta(\sigma_i)$	Set of the data signals communicated through the shared resource σ_i. Please note that this function returns a set, not just one data signal. This is because in the case of a one-to-many communication, all the data signals can be communicated through the same shared resource.
$\zeta'(s_i)$	Shared resource corresponding to the signal s_i
Ω_{r_i}	Set of all the shared resources accessed by the input/output ports of the runnable r_i
$\omega_{p^{in}_{r_i,j}}/\omega_{p^{out}_{r_i,k}}$	WCET of the runnable entity r_i on the critical section used for accessing shared resources. The access is through the input/output port $p^{in}_{r_i,j}/p^{out}_{r_i,k}$
$idx(r_i)$	In the context of the time triggered systems this work accounts for an order of runnables inside a task. Runnables from different paths can be partitioned in the same task and hence their ordering influences the response times of paths (see section 4.2). This in fact constitutes an additional, fourth dimension of the deployment problem which is considered by the proposed technique (see subsection 4.4.5). Partitioning of runnables belonging to different transactions (data driven activation) is not permitted, consequently for the data driven activation model order is not considered. The index of a runnable inside a task is $j = idx(r_i)$. The $r_{i,j}$ means that runnable r_i is at the j^{th} position in a task.

Table 4.3. Additional Concepts for the Time Driven Activation Model

In the time driven activation model each runnable entity r_i is triggered by a periodic timer event. Therefore the runnables are independent in a sense that the triggering of each doesn't depend on the other runnables.

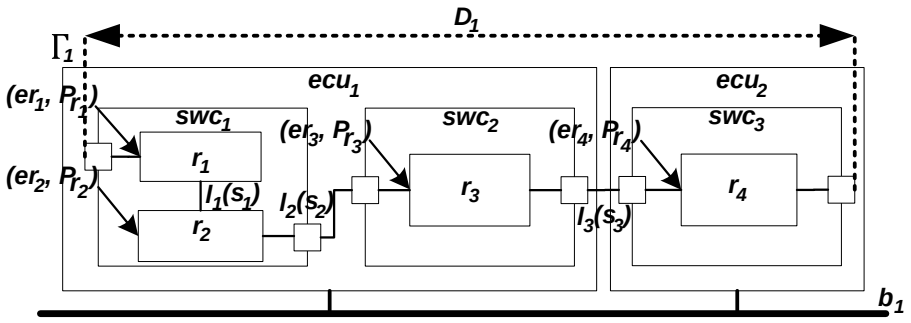

Figure 4.2. Time Driven Activation Model

Characteristic for this activation model is communication between runnables that needs to be protected due to the concurrent access between writers and readers but also due to the necessity of preserving the semantic of a functional model (defined through the runnables and exchanged data signals). The last concern is called flow preservation and requires that the readers consume the right instances of data sent by the writers. This is not the case for the data driven communication as when the data is sent by the writer it is immediately consumed by the reader which in fact is triggered by the data arrival. Therefore the protection mechanism has to be specified for each data signal communicated between runnables of different tasks that are deployed on the same ECU. There exist two mechanisms which can be used for data protection. Our technique for the deployment of the time triggered systems (see subsection 4.4.5) will operate a choice between a time-consuming and a memory-consuming protection mechanism.

4.2. Schedulability Analysis

Timing analysis concerns computation of the response times for runnables and global signals and also computation of end-to-end responses. Schedulability analysis test differs among the two activation models. This implied their description within two separate subsections.

4.2.1. Schedulability Analysis for DD

Schedulability analysis for data-driven activation model comes from [34] and is called holistic analysis. The main rationale behind the choice of this schedulability test was a possibility to formulate it using MILP (Mixed Integer Linear Programming). The MILP was used in the evaluation stage (see 4.4.4, 4.4.6, 4.5.3) to look for optimal solutions to which results of proposed optimization technique could have been compared. The MILP formulation for more recent and more exact schedulability test like [35] was not possible.

Schedulability of Runnable Entities – to compute the WCRT of runnables this work uses the response time analysis with jitter propagation given in [34]. The last operates on the OS tasks whereas in this work it is adapted to consider the runnable entities. The WCRT R_{r_i} of a runnable entity r_i is computed by considering all the q runnable instances (distinct executions of r_i after activation) in the busy period, as follows in the equation 4.1.

$$R_{r_i} = \max_{q=1,2,\dots} \left[W_{r_i}^{(q)} - (q-1)P_{r_i} + J_{r_i} \right] \quad \textbf{4.1}$$

Since r_i is executed by a task, its release jitter J_{r_i} is the task release jitter (see 4.2), that is, the largest among all the latest release times for the runnables in the same task, which is zero if the runnable has no predecessor (or a predecessor within the same task), or the worst case response time of the signal it receives from a remote predecessor runnable (s_i is the signal received by r_i). The $W_{r_i}^{(q)}$ is the completion time of runnable r_i and is computed according to 4.3. The function $hp(r_i)$ will return all the runnables allocated on the same ECU with a priority higher than that of the runnable r_i. The last part of the equation 4.3 represents the preemption time from functions belonging to $hp(r_i)$. When a task contains runnables with different periods, its interference is computed as the sum of the interferences of its runnables. The completion time is computed for $q = 1,2,\dots$ until the busy period ends, that is, an instance completes at or before the activation of the next instance.

$$J_{r_i} = \max_{\forall r_j \in R \tau(r_i)=\tau(r_j)} \{0, R_{s_i}\} \quad \textbf{4.2}$$

$$W_{r_i}^{(q)} = q\left(C_{r_i} + \sum_{r_j:\tau(r_i)=\tau(r_j)} C_{r_j} \right) + \sum_{r_j \in hp(r_i)} \left\lceil \frac{W_{r_i}^{(q)} + J_{r_j}}{P_{r_j}} \right\rceil C_{r_j} \quad \textbf{4.3}$$

Schedulability of Signals – R_{s_i} is computed only for signals representing an inter-ECU communication, otherwise it equals to 0. Its computation uses a formula similar to that for computing WCRT of a runnable (see 4.4). The difference is in the additional term that represents the blocking time B_{s_i} which needs to be added to the WCRT. It is an effect of impossibility to

preempt bus message while it is being transmitted on the bus, even if it has lower priority. This property applies to the CAN bus, considered in this work as a communication medium. According to 4.6 the blocking time for signal s_i equals to the maximal WCTT among all the messages transferred on the same bus and with a lower priority. The release jitter of a remote signal (see eq. 4.7) is the worst-case response time of its sender runnable represented as $Snd(s_i)$.

$$R_{s_i} = \max_{q=1,2,\dots} \left[W_{s_i}^{(q)} - (q-1)P_{s_i} + J_{s_i} \right] \quad \textbf{4.4}$$

$$W_{s_i}^{(q)} = B_{s_i} + q\left(C_{s_i} + \sum_{s_j : m(s_i)=m(s_j)} C_{s_i} \right) + \sum_{s_j \in hp(s_i)} \left\lceil \frac{W_{s_i}^{(q)} + J_{s_j}}{P_{s_j}} \right\rceil C_{s_j} \quad \textbf{4.5}$$

$$B_{s_i} = \max_{\forall m_j \in M\, b(m_j)=b(s_i) \wedge \Pi_{m_j} < \Pi_{s_i}} C_{m_j} \quad \textbf{4.6}$$

$$J_{s_i} = J_{Snd(s_i)} \quad \textbf{4.7}$$

End-to-end Responses Computation – the response time R_i of a transaction Γ_i equals to the response time of the last runnable entity in this transaction.

4.2.2. Schedulability Analysis for TD

This analysis is based on the work of [36] and adapted to consider runnable entities. Adaptation is due to the fact that the entities considered in the analysis of [36] focus on OS tasks and doesn't consider functional entities as in our case.

Schedulability of Runnable Entities - worst case response time of a runnable r_i, for which $idx(r_i) = j$, is represented with $R_{r_{i,j}}$ and computed according to 4.8. The $C_{\tau_{i,j}}$ (see eq. 4.9) is the worst case computation time of the task until the j^{th} runnable partitioned in this task. Please note that the partitioning of the runnables with the harmonic periods in the same task is allowed. This means that when the task is executed not all of the runnables will be activated. Therefore the $C_{\tau_{i,j}}$ varies. However the worst case scenario is assumed hence this work accounts for all the

runnables up till the j^{th} when computing the $C_{\tau_{i,j}}$. The B_{τ_i} is a blocking time of a task τ_i. Blocking time depends on the shared resources accessed by the task and the way in which the shared resources are protected from multiple accesses. If the shared resource is protected with a semaphore lock, it causes a blocking time. The semaphore lock in our case is realized through the Priority Ceiling Protocol (PCP) [37]. The same blocking time applies to all the runnables that are partitioned in the same task and therefore it is computed for a task. To compute the blocking time with the PCP few additional things have to be clarified. First, the shared resources of a task τ_i are specified with the set $\Omega_{\tau_i} = \bigcup_{\tau(r_j)=\tau} \Omega_{r_j}$. This means that the task inherits the access to the shared resources from the runnables partitioned in this task. The WCET of a task τ_i for accessing (reading/writing) a critical section of a shared resource σ_i is represented with $\tau_i\left(\omega_{\sigma_i}^{in}\right) = \max_{r_i}(\omega_{p_{r_i,j}^{in}})$ / $\tau_i\left(\omega_{\sigma_i}^{out}\right) = \max_{r_i}(\omega_{p_{r_i,j}^{out}})$. Function $hp(r_i)$ returns all the runnable entities allocated on the same ECU as r_i, with the priority higher than r_i.

$$R_{r_{i,j}} = B_{\tau(r_i)} + C_{\tau(r_i),j} + \sum_{r_k \in hp(r_i)} \left\lceil \frac{R_{r_{i,j}}}{P_{r_k}} \right\rceil C_{r_k} \qquad \textbf{4.8}$$

$$C_{\tau_{i,j}} = \sum_{\tau(r_k)=\tau_i \wedge k \leq j} C_{r_k} \qquad \textbf{4.9}$$

Schedulability of Signals - Worst case response time for a signal is computed in case when s_i represents inter-ECU communication (see eq. 4.10). Otherwise the response time equals 0. This work just as for the data driven model considers the CAN bus to communicate distributed runnables. Therefore the computation of R_{s_i} accounts for a blocking time B_{s_i} which results from the impossibility to preempt a message that is being already transmitted on the bus, even if it has lower priority. Its computation was explained in the context of schedulability analysis of signals for data driven activation model (see eq. 4.6) and remains the same here. Function $hp(m_i)$ returns all the messages of the same bus as m_i with a priority higher.

$$R_{s_i} = B_{s_i} + C_{m(s_i)} + \sum_{m_k \in hp(m(s_i))} \left\lceil \frac{R_{s_i}}{P_{s_i}} \right\rceil C_{m_k} \qquad \textbf{4.10}$$

End-to-end Responses Computation – this is a computation of a response time of a path. The worst case end-to-end latency R_{Γ_i} is computed for each path Γ_i by adding the worst case response times of all the runnables and global signals (i.e. signals representing inter-ECU communication), as well as the periods of all the global signals and their reader runnables on the path (see (8)). Set Φ represents all the global signals. The $read_{s_k,\Gamma_i}$ is the reader runnable of s_k on the specific path Γ_i and $P_{read_{s_k,\Gamma_i}}$ represents its period.

$$R_{\Gamma_i} = \sum_{r_j \in \Gamma_i} R_{r_j} + \sum_{s_k \in \Gamma_i \wedge s_k \in \Phi} (R_{s_k} + P_{s_k} + P_{read_{s_k,\Gamma_i}}) \qquad \textbf{4.11}$$

4.3. Refinement of the EAST-ADL2/AUTOSAR Methodology

The section 2.5 and the Figure 2.5 from page 55 present the EAST-ADL2/AUTOSAR methodology. The change advertised in this section concerns distribution of responsibilities between two levels, functional level covered by the EAST-ADL2 and the implementation level covered by the AUTOSAR. The *Design* activity which is done at the functional level includes allocation step in which atomic functions are allocated on the ECUs. This determines the allocation of runnable entities due to the assumption in which runnable entities are transformed from the atomic functions. The mapping between these two modeling concepts is specified in the subsection 2.9. It is a result of their semantic proximity. The AUTOSAR level needs to respect this allocation decision by simply reflecting it in the runnables' allocation. This infers that the deployment problem cannot be holistically considered at the AUTOSAR level as one problem dimension, i.e. the allocation is already fixed. Therefore a deployment technique can operate only on two instead of three parameters, i.e. the partitioning and the priorities assignment. This significantly shrinks the design space to explore and hence might exclude optimal deployment configurations. A sound workaround would be to perform an entire deployment at the EAST-ADL2 level. This is however impossible as partitioning and priorities assignment cannot be done at the EAST-ADL2 level due to the missing concepts. This language is intended to abstract from the implementation objectives hence the tasks modeling is out of its scope. Consequently the overall methodology has to be changed. This work advocates the change in which the allocation is postponed till the implementation level. The gain of this change will be assessed in the section

4.6 by comparing the deployment techniques adjusted to the old methodology and those compliant with the refined methodology.

4.4. Deployment

The main objective of a deployment whether for DD or TD is the integration of software architecture with a hardware platform. The software architecture is represented by software components embedding communicating runnable entities which send data signals. Hardware platform contains ECUs and BUSes. This section presents the deployment techniques that can be run at the AUTOSAR level of the refined EAST-ADL2/AUTOSAR methodology in order to support the deployment process. There are two separate techniques adjusted to support the data-driven and time-driven semantic of execution. The subsection 4.4.1 formalizes the deployment problem correspondingly for data-driven and time-driven activation models. It presents the main objectives driving the deployment and a set of constraints designating a correct deployment. Next is the presentation of a related work. Subsections 4.4.3 and 4.4.5 specify the techniques for the deployment. Their evaluation assessing the quality of obtained results and the scalability is done correspondingly in 4.4.4 and 4.4.6.

4.4.1. Formalization of Deployment

Deployment for DD

The formalization of a deployment consists of three steps. First is the definition of a problem itself, second these are the objectives that drive the deployment process and finally it is a specification of constraints that need to be respected by the final, deployed architecture.

Problem Formulation – the goal of a deployment is to:

1) Assign the runnables from the set R to the elements of the set E which contains ECUs.
2) Assign the signals from the set S to the elements of the set B which contains BUSes.
3) Group the runnables from the set R to the tasks and hence define the set T.
4) Group the signals from the set S to the messages and hence define the set M.
5) Assign the priorities to the OS tasks.
6) Assign the priorities to the messages.

The steps 1) and 2) are called allocation, steps 3) and 4) this is the partitioning and 5) and 6) the scheduling.

Optimization Metrics – optimization metric can be defined based on the system requirements. In the context of the DD, this work accounts for two formulations aiming to optimize the same objective, i.e. end-to-end responses. This function is called $f_{e2e}(\Psi_i)$ and can have one of the two forms:

1) The minimization of the sum of all (or some) transactions latencies. It is a loose indication of the system performance.

$$\min \sum_{\Gamma_i} R_i \qquad \textbf{4.12}$$

2) The maximization of the minimum transactional slack time. A slack time for a given transaction is defined as the difference between the deadline and latency of the transaction. This metric can be related to the concept of robustness (or extensibility) of the system against changes in the time parameters of some runnables.

$$\max \min_{\Gamma_i}[\mathrm{D}_i - \mathrm{R}_i] \qquad \textbf{4.13}$$

Deployment Constraints – the deployment process has to respect multiple constraints.

1) Allocation Constraints – these constraints concern allocation and resources utilization constraints.
    a. Each runnable can be allocated only on one ECU. If we denote with $SE(r_i)$ set of ECUs on which runnable entity is allocated the constraint would be of that presented under 4.14.

$$\bigwedge_{r_i \in R} |SE(r_i)| = 1 \qquad \textbf{4.14}$$

    b. Similar constraint as in a. applies to the signals, i.e. each signal that is global, i.e. represents inter-ECU communication. If we denote with $SB(s_i)$ set of BUSes on which data signal is allocated the constraint would be of that presented under 4.15.

$$\bigwedge_{s_i \in S \wedge s_i \in \Phi} |SB(s_i)| = 1 \qquad \textbf{4.15}$$

    c. Fixed Allocation – for certain runnables allocation is constrained to the subset of all the ECUs.
    d. Two communicating runnables cannot be allocated on separate ECUs for which there is no BUS connection. Function $\sigma(e_i, e_j)$ will return true if there is a

connection between the ECU e_i and e_j. The $src(l_i)$ is a source runnable of a link l_i and $dst(l_i)$ represents its destination runnable.

$$\bigwedge_{r_i,r_j \in R} E(r_i) \neq E(r_j) \wedge (src(l_k) = r_i \wedge dst(l_k) = r_j) \Longrightarrow \sigma(E(r_i), E(r_j)) \quad \textbf{4.16}$$

e. Utilization Constraint – the allocation cannot exceed utilization threshold specified for each ECU/BUS. The $U_{e_i}^{ecu}$ and $U_{b_i}^{bus}$ is the maximal utilization constraint specified for the ECU e_i and the BUS b_i. Values of $U_{e_i}^{ecu}$ and $U_{b_i}^{bus}$ are not greater than one.

$$\bigwedge_{e_i \in E} \sum_{E(r_j)=e_i} \frac{C_{r_j}}{P_{r_j}} \leq U_{e_i}^{ecu} \quad \textbf{4.17}$$

$$\bigwedge_{b_i \in B} \sum_{B(s_j)=b_i} \frac{C_{s_j}}{P_{s_j}} \leq U_{b_i}^{bus} \quad \textbf{4.18}$$

2) Partitioning Constraints – these constraints concern allocation and resources utilization constraints.

a. Harmonic Rate – this constraint forbids the partitioning of two runnables/signals with non-harmonic periods on the same task/message.

$$\bigwedge_{r_i,r_j \in R} P_{r_i} \equiv P_{r_j} \neq 0 \wedge P_{r_j} \equiv P_{r_i} \neq 0 \Longrightarrow \tau(r_i) \neq \tau(r_j) \quad \textbf{4.19}$$

$$\bigwedge_{s_i,s_j \in S} P_{s_i} \equiv P_{s_j} \neq 0 \wedge P_{s_j} \equiv P_{s_i} \neq 0 \Longrightarrow m(s_i) \neq m(s_j) \quad \textbf{4.20}$$

b. Each runnable can be partitioned only in one task. If we denote with $ST(r_i)$ set of tasks in which runnable entity r_i is partitioned the constraint would be of that presented under 4.21.

$$\bigwedge_{r_i \in R} |ST(r_i)| = 1 \quad \textbf{4.21}$$

c. Similarly as in b, each signal can be partitioned only in one message. If we denote with $SM(s_i)$ set of messages in which signal s_i is partitioned the constraint would be of that presented under

$$\bigwedge_{s_i \in S} |SM(s_i)| = 1 \quad \textbf{4.22}$$

3) Execution Order Constraints

   a. Local Total Order – this constraint prevents from having two tasks/messages of the same ECU/BUS being assigned the same priority.

$$\bigwedge_{\tau_i, \tau_j \in T} E(\tau_i) = E(\tau_j) \Rightarrow \pi_{\tau_i} \neq \pi_{\tau_j} \quad \textbf{4.23}$$

$$\bigwedge_{m_i, m_j \in S} B(m_i) = B(m_j) \Rightarrow \pi_{m_i} \neq \pi_{m_j} \quad \textbf{4.24}$$

   b. Runnables Order – order of execution of the runnable entities influences the priorities assignment for tasks. Namely if the runnables r_i and r_j belong to the same transaction and r_i precedes in the execution runnable r_j then runnable r_i should be partitioned in the task with a priority higher or equal to the priority of a task hosting the runnable r_j.

$$\bigwedge_{r_i, r_j \in R} \Gamma(r_i) = \Gamma(r_j) \wedge E(r_i) = E(r_j) \wedge i < j \Rightarrow \pi_{\tau(r_i)} \geq \pi_{\tau(r_j)} \quad \textbf{4.25}$$

   c. Signals Order – this constraint is analogous to the runnables order constraint.

$$\bigwedge_{s_i, s_j \in S} \Gamma(s_i) = \Gamma(s_j) \wedge B(s_i) = B(s_j) \wedge i < j \Rightarrow \pi_{m(s_i)} \geq \pi_{m(s_j)} \quad \textbf{4.26}$$

4) Latency Constraint – response time of each transaction should be within a predefined deadline.

$$\bigwedge_{\Gamma_i \in \xi} R_i \leq D_i \quad \textbf{4.27}$$

Deployment for TD

The formalization of a deployment for the TD follows the same principles as for the DD.

Problem Formulation – the deployment problem in the context of the TD is more complex due to the additional dimension, which is ordering in addition to the allocation, partitioning and scheduling. The goal in this case is to:

1) Assign the runnables from the set R to the elements of the set E which contains ECUs.

2) Assign the signals from the set S to the elements of the set B which contains BUSes.
3) Group the runnables from the set R to the tasks and hence define the set T.
4) Group the signals from the set S to the messages and hence define the set M.
5) Order the runnables inside the tasks, i.e. for each runnable r_i define its index $j = idx(\mathrm{r}_i)$ inside a task.
6) Assign the priorities to the OS tasks.
7) Assign the priorities to the messages.
8) Assign the protection mechanism for each shared resource σ_i from the set Ω. Function $\gamma(\sigma_i)$ will return the value representing the protection mechanism used to protect a shared resource σ_i.

Concerning the point 5) the position of a runnable inside a task has an impact on its response time and implicitly on a response time of a path or paths to which this runnable belongs. The computation of a response time $R_{r_{i,j}}$ for a runnable r_i at the position j (see eq. 4.8) contains $C_{r_{i,j}}$ which increases with every preceding runnable, by its WCET. Consequently the WCRT will increase as well.

Concerning the point 8) protection mechanism has to be specified for each signal communicated between runnables of different tasks that are deployed on the same ECU. This is due to the asynchronous communication between periodic runnables and hence, mechanism to provide the data consistency is necessary. This work considers two mechanisms:

- Wait-free access method such as Rate Transition (RT) block [38] – this mechanism behaves like a Zero-Order Hold block or a Unit Delay block plus a Hold block or Sample and Hold (for slow to fast transitions). Its implementation consists of a switched buffer. This mechanism incurs negligible time overhead but it consumes additional memory.
- Semaphore Lock (SL) – this work uses semaphore locks based on the immediate priority ceiling protocol. Priority of a runnable that is accessing a shared resource is raised to the ceiling priority of a resource. The SL, opposite to the RT, imposes no additional memory overhead, however it suffers timing delays in the form of a blocking time.

In this case function $\gamma(\sigma_i)$ will return one of the two possible values representing a protection mechanism used to protect a shared resource σ_i. Value SL concerns semaphore lock, whereas RT

means Rate Transition block. Overall memory overhead M_{e_i} for ECU e_i is computed according to 4.28.

$$M_{e_i} = \sum_{e(r_j)=e_i} M_{r_j} + \sum_{e(\sigma_k)=e_i} M_{\sigma_k} \quad \textbf{4.28}$$

The M_{σ_k} is a memory overhead caused by the RT and is computed according to [39] (see eq. 4.30). For this additional notation is defined. The set of readers with higher (lower) priority than the writer $r^w(\sigma_i)$ are denoted as $r^{HR}(\sigma_i)$ ($r^{LR}(\sigma_i)$). Function $DS(S_i)$ returns the size of the data signals in the set S_i. As specified before in the Table 4.3 function $\zeta(\sigma_k)$ returns the set of data signals communicated through the shared resource σ_k. The formula 4.30 is a simplification of what is included in [39] as in this work preemption thresholds are not considered.

$$M_{\sigma_k} = DS\big(\zeta(\sigma_k)\big) n_{\sigma_k} \quad \textbf{4.29}$$

$$n_{\sigma_k} = \begin{cases} \sum_{r_i \in r^w(\sigma_k), r_j \in r^{LR}(\sigma_k)} w_{\sigma_k} + 2w_{\sigma_k} \; if r^{HR}(\sigma_k) \neq \emptyset \\ \sum_{r_i \in r^w(\sigma_k), r_j \in r^{LR}(\sigma_k)} w_{\sigma_k} + w_{\sigma_k} \; if r^{HR}(\sigma_k) = \emptyset \end{cases} \quad \textbf{4.30}$$

$$w_{\sigma_k} = \begin{cases} 1 \; if \; \gamma(\sigma_i) = RT \\ 0 \; if \; \gamma(\sigma_i) = SL \end{cases} \quad \textbf{4.31}$$

Below table concludes the new notation.

Concept	Definition
$\gamma(\sigma_i)$	Function returning the protection mechanism specified for the shared resource σ_i
RT	Value returned by the function γ if the protection mechanism this is Rate Transition block
SL	Value returned by the function γ if the protection mechanism this Semaphore Lock

M_{e_i}	Memory requirement for the ECU e_i
M_{r_i,e_j}	Stack memory usage of a runnable r_i on the ECU e_j
M_{σ_k}	Memory overhead caused by the Rate Transition block used to protect shared resource
$r^{HR}(\sigma_i)$	Set of reader runnables of a shared resource σ_i with a priority higher than the writer
$r^{LR}(\sigma_i)$	Set of reader runnables of a shared resource σ_i with a priority lower than the writer
$DS(S_i)$	Size of the data signals from the set S_i

Table 4.4. Additional Notation for TD

Optimization Metrics – the synthesis process is driven by the predefined optimization criteria. This work defines two optimization metrics and for each its importance can be specified by assigning a weight. Therefore our final fitness function $f(\Psi_i)$ where Ψ_i represents a final configuration, i.e. deployed architecture, is a weighted sum of two functions as in the equation 4.32.

$$f(\Psi_i) = w_1 \mathrm{f}_{e2e}(\Psi_i) + w_2 \mathrm{f}_m(\Psi_i) \qquad \textbf{4.32}$$

The two functions f_{e2e} and $\mathrm{f}_m(\Psi_i)$ impose the optimization of the end-to-end response times and the memory.

1) End-to-end Responses Optimization – optimization of the end-to-end responses aims at minimizing the response times of paths, relatively to their deadlines (see eq. 4.33). Their optimization serves to improve the system performance.

$$\mathrm{f}_{e2e}(\Psi_i) = |\xi| - \sum_{\Gamma_j} \frac{R_{\Gamma_j}}{\mathrm{D}_{\Gamma_j}} \qquad \textbf{4.33}$$

2) Memory Optimization - Optimization of memory (see eq. 4.34) aims at minimizing the additional memory overhead that can be caused by using the Rate Transition blocks and inappropriate balancing when placing runnables on the ECUs. The last is due to the heterogeneous nature of the ECUs. The $M_{e_i}^{max}$ represents the worst case possible memory overhead caused for e_i; $M_{e_i}^{max} = \sum_{r_j} M_{r_j,e_i} + \sum_{\sigma_k} M_{\sigma_k}$. Its computation assumes that each runnable is partitioned in one task, writer has always higher priority than all its readers and all shared resources are protected with the RT.

$$\mathrm{f}_m(\Psi_i) = |E| - \sum_{e_j} \frac{M_{e_j}}{M_{e_j}^{max}} \quad \textbf{4.34}$$

Deployment Constraints – the deployment for TD should respect the same set of constraints as specified for DD (see subsection 4.4.1). There are few additional constraints presented below.

1) Indexes – specification of the ordering through the assignment of the indexes representing the position of a runnable inside a task should refer to the position of the runnables within a path. Namely if there are two runnables of the same path and runnable r_i precedes runnable r_j then the index assigned for r_i should be smaller than this of r_j.

$$\bigwedge_{r_i, r_j \in \mathrm{R}} \Gamma(r_i) = \Gamma(r_j) \wedge \tau(r_i) = \tau(r_j) \wedge i < j \Rightarrow idx(r_i) < idx(r_j) \quad \textbf{4.35}$$

2) Shared Resource – each data signal communicated between the runnables of the same ECU but partitioned in different tasks, needs to be communicated through the shared resource.

$$\bigwedge_{s_i \in \mathrm{S}} E\big(Snd(s_i)\big) = E\big(Rcv(s_i)\big) \wedge \tau\big(Snd(s_i)\big) \neq \tau\big(Rcv(s_i)\big) \Rightarrow \bigvee_{\sigma_j} s_i \in \zeta(\sigma_j) \quad \textbf{4.36}$$

3) Protection Mechanism – for each shared resource one of the two possible protection mechanisms needs to be specified.

$$\bigwedge_{\sigma_i \in \Omega} \gamma(\sigma_i) = RT \vee \gamma(\sigma_i) = SL \quad \textbf{4.37}$$

4.4.2. Related Work

The literature on the synthesis and in particular, deployment is rich. It can be structured according to multiple criteria like the type of systems considered, optimization constraints, domain, and so forth. The main criteria to structure this survey over the related work are based on the activation semantics, i.e. whether it is data or time driven, parameters to manipulate during the deployment and optimization objectives.

In order to find good solutions to deployment problems optimization techniques have been extensively used. A very good survey provided in [40] classifies 188 papers along multiple criteria such as design goals, dimensionality (single objective versus multi-objective), domain (embedded systems, enterprises systems, etc.), phase (architecture versus run-time optimization), types of constraints, architecture representation (ADL, UML, etc.), optimization strategy (e.g.

exact vs metaheuristics, etc.) and constraint handling (prohibition, penalty, etc.). What is however significant is that none of the surveyed papers treats the worst-case latency of the deployed transactions/paths as either design constraint or goal.

The concept of end-to-end deadline has been discussed in many research works. This applies both to single-processor and distributed architectures. In particular, for data-driven activation models end-to-end deadlines were considered in the context of schedulability analysis test such as holistic analysis with jitter propagation used in this work [34], or model with offsets as in [35]). Timing analysis techniques advanced significantly, considering new activation models, communication protocols or more expressive tasks representations (e.g. digraph model [41]). The optimization of deployment has not received comparable attention. [42] and [43] proposes a heuristics-based design optimization algorithm for mixed time-triggered and event-triggered systems. Its main assumption is that the nodes (in our case ECUs) are synchronized. An integrated framework for optimization is proposed in [44] for systems with periodic tasks on a network of processor nodes connected by a time-triggered bus. Authors use Simulated Annealing (SA) combined with geometric programming to hierarchically explore task allocation and assignment of tasks' priority and period. In [45] the process of allocation of tasks and priority assignment targets the optimization of system flexibility, i.e. ability to adapt to changes which is important for real-time systems. The possible change this is introduction of new tasks into the system which obviously impacts the response-times of already deployed tasks. To solve the problem, just as in the previous work, authors are using simulated annealing. Work of Hamann et al. [46] optimizes multi-dimensional robustness criteria in a complex embedded system. Their approach is based on the stochastic multi-dimensional sensitivity analysis technique. Authors consider multiple problems affecting system performance such as changes in the execution times of tasks but also period speed-ups, etc. Azketa et al. [47] delivers an approach based on the genetic algorithms that optimizes the assignment of priorities to tasks and messages and then it maps them on the execution platform. Similarly allocation and scheduling decisions are being optimized in [48] and [49] under the real-time constraints. There are also approaches which consider only mono-processor architectures such as [50], [51] or [52] hence for all of them allocation is out of scope.

The Table 4.5 gathers all the discussed work closely related to the DD activation model and with a similar goal of optimizing the deployment. **It can be concluded that none of these works**

targets all the possible deployment decisions at once, i.e. allocation, partitioning and scheduling.

Work	Allocation	Partitioning	Scheduling	Optimization of End-to-End Responses
Pop [42], [43]		✓	✓	
He [44]	✓		✓	✓
Bate [45]	✓		✓	✓
Hamann [46]			✓	
Azketa [47]	✓		✓	✓
Kugele [48]	✓		✓	✓
Richard [49]	✓		✓	✓
Bartolini [50]		✓	✓	✓
Saksena [51]		✓	✓	✓
Kodase [52]		✓	✓	✓
Proposed Approach	✓	✓	✓	✓

Table 4.5. Summary of the Related Work for DD

Concerning the time driven activation model and the optimization objectives, the approach presented in this thesis is closely related to the following works [53], [54]. The [53] and [36] similarly as in our approach consider periodic activation and end-to-end responses as optimization criteria. The main difference is that the authors are considering OS tasks as an allocation unit and hence partitioning and ordering is fixed for them. Ferrari et al. [55] is the first work discussing possible strategies to protect shared data items and memory/timing tradeoffs. The work in [56] proposes a two-step technique for the allocation of AUTOSAR software components to the ECUs, taking into account protection mechanism as a parameter to specify. However it considers neither partitioning nor ordering. Authors of [39] and [54] also relate to the periodic runnables in their model. They consider additional mechanisms that can assure data consistency like the absence of preemption. The last can be done by defining so called preemption thresholds or preemption groups. In their work the allocation is fixed and hence their

approach is for local optimization. Interestingly, the order of runnables as the parameter to manipulate is considered. The Table 4.6 groups the related work and highlights their main features. **As can be seen there is no work supporting the allocation, partitioning, scheduling, ordering and memory protection specification which is the desired goal of a technique proposed in this thesis.** In addition only [54] treats the deployment in regards to the optimization of the end-to-end responses.

Work	Allocation	Partitioning	Scheduling	Ordering	Memory Protection	Optimization of End-to-End Responses
Zhu [53]	✓		✓			
Zhu [36]	✓		✓			
Ferrari [55]					✓	
Zhang [56]	✓	✓[2]			✓	
Zeng [39]		✓	✓	✓	✓	
Zeng [54]				✓	✓	✓
Proposed Approach	✓	✓	✓	✓	✓	✓

Table 4.6. Summary of the Related Work for TD

4.4.3. Technique for Optimized Deployment of DD

The technique for the deployment of the DD is based on the genetic algorithms (GA). Genetic algorithm is an optimization technique patterned after natural selection in biological evolution. Algorithm 1 is a general form of a genetic algorithm. In a GA, the space of all possible solutions (feasible and not feasible) to the optimization problem is encoded using a string of bits, called chromosome. Each bit or group of bits in the sequence typically encodes one parameter of the solution (such as the placement of a runnable entity or the priority of a task). Several solutions are generated at each round (population), starting from an initial set and then obtaining new

[2] Partitioning uses predefined strategy, i.e. all the runnables with the same period are partitioned in the same task.

solutions by a composition function (or *crossover*) that applies to two chromosomes and produces a new one or by a mutation operator that changes the bit string of a chromosome to generate a new one. Each new generation (or offspring) is evaluated. Some bit strings correspond to non-feasible solutions, or dead individuals and are discarded. A set of the most promising ones is retained and used for computing the next generation. Each problem intended to be solved with the GA requires the definition of an encoding, crossover, mutation operator and fitness function. Encoding is what mostly differs among the problems. The same crossover or mutation operators can be used for different problems but in many cases their proper choice can have a significant impact on a capability of the GA to deliver an optimal result. In the consecutive paragraphs, this subsection introduces the specification of an encoding, way to generate initial population, crossover and mutation operators, as well as the *correction mechanism* which maintains the correctness of the chromosomes.

Algorithm 1: GA

```
1: // Define encoding, crossover, mutation operator and fitness function.
2: // Specify the size of an initial population - P_size
3: Generate initial population
4: while termination condition is not met do
5:     Evaluate each solution from the population P
6:     Generate new population P by applying the crossover and mutation operators
7: end while
8: return the best solution from P
```

Algorithm 1. General Form of Genetic Algorithm

Encoding – the encoding definition translates a solution configuration in a string of bits. In the placement problem, a specific solution, i.e. a single chromosome ch_i, represents a specific deployment configuration, i.e. allocation of runnables/signals to ECUs/BUSes, partitioning of runnables/signals to tasks/messages and assignment of priorities to tasks and messages. This work uses the value encoding, in which each gene g_i (subset of bits) in a chromosome contains a specific value. In this case, a gene relates either to a runnable entity or a data signal. For the first, gene $g_i = ch_j(r_k)$ stores the value $V(g_i)$ representing runnable's allocation and partitioning. For a data signal, value stored depends whether it is a global data signal or a data signal that is communicated locally. Value for a global data signal will hold information about the BUS and the message in which it is partitioned. If this is a local data signal s_i the value will not have any meaning, as in case of the intra-ECU communication, signals are communicated through the

message passing and they have to be assigned neither to bus nor to message. This whether the s_i is a global or local data signal depends on the values assigned for the genes relating to the runnable entities which exchange the s_i. If these genes hold the same value representing ECU then the signal is local, otherwise it is global.

The gene value $V_{r_j}(g_i)$ for the runnable r_j is one number but stores information about the ECU number on which runnable r_j is allocated and the task number in which it is partitioned. The $V_{r_j}(g_i)$ for runnable r_j for which selected ECU is e_k and the task τ_l is computed in a specific way, according to 4.38. The max_{ECU} is computed as a maximal number of runnables that can be hosted by one ECU without violation of utilization (for this WCETs and periods of runnables are used).

$$V_{r_j}(g_i) = (k-1) * max_{ECU} + (l-1) \qquad \textbf{4.38}$$

The gene value for a data signal, if transmitted on the bus, is computed in a similar way (see 4.39). The max_{BUS} is calculated as a maximal number of signals that can be hosted by one BUS without violation of utilization (for this WCTTs and periods of signals are used). Figure 4.3 presents an example of a chromosome for a specific deployment configuration. For example the $V_{r_2}(g_3)$ equals 2 which means that r_2 is allocated on the ECU 1 and partitioned in task with index 3 (this index defines also a priority of a task). Signal s_1 is a local signal therefore value of its gene is specified as x, to show that it has no meaning.

$$V_{s_j}(g_i) = (k-1) * max_{BUS} + (l-1) \qquad \textbf{4.39}$$

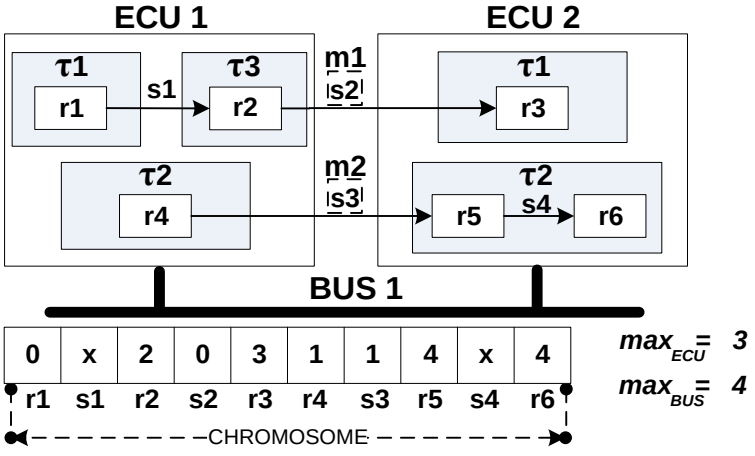

Figure 4.3. Example of a Chromosome for a Specific Deployment Configuration

Initial Population - the initial population is generated randomly but to generate correct chromosomes, possible range of values for each gene depends on the values already assigned to others. Correct means chromosome representing deployment configuration that respects the constraints formalized in the subsection 4.4.1. The number of the elements in the initial population has a high impact on the quality of results obtained but a negative influence on the runtime. The GA_{pop} will represent a number of elements in the initial population.

Fitness Function – defines how much the solution optimizes the performance criteria. Chromosomes are ranked according to this function and, the higher the rank, the higher the probability that the chromosome is selected as a parent for a crossover or the target of a mutation. The fitness function used by this specification of the GA was specified in subsection 4.4.1 and equations 4.12 and 4.13.

Evolution - the evolution of a population is through the selection of chromosomes with good fitness and applying the crossover and mutation mechanism on them. The selection of the crossover operator is very important for the quality of the GA solution. The operator combines information from two parent chromosomes. The selection of parents can be done in many ways, but it is always highly dependent on a chromosome fitness rate. This work uses the OX3 crossover operator [57] with a tournament selector [58] (with size equal to 5). The tournament selector with size 5 will first create two sets with 5 randomly chosen chromosomes. The most fit chromosome from each set will be taken and these two chromosomes will be used as parents for the crossover. Then the OX3 operator will randomly select the "crossover points", i.e. indexes of genes that will constitute the boundaries of the crossover operation. The values between these points are copied from the first/second parent to the second/first child in the same absolute position. The remaining values are copied from the first/second parent to the first/second child. A simple result of the application of this operator on two random chromosomes is shown on the Figure 4.4.

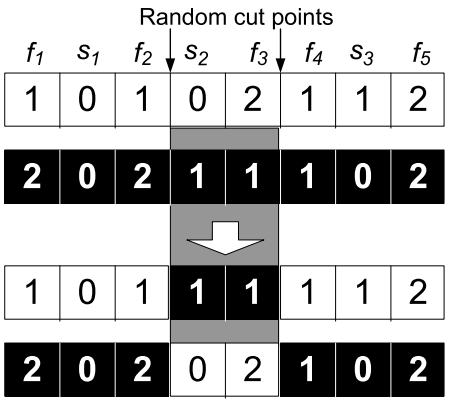

Figure 4.4. OX3 Crossover Operator

The mutation operator chooses a random point in a chromosome and changes the value of the gene at the selected point to a new random value. If the randomly selected gene corresponds to a runnable, the new value is chosen from the list of available execution nodes and tasks. If it relates to a message, the new value is chosen from the list of available buses. Additionally the probability that a chromosome after selection will undergo mutation is 70% which is a configuration parameter. Finally if the fitness value of a mutated chromosome is worse than before, the mutation is rolled back.

Correction Mechanism – is used to avoid the generation of non-feasible solutions that can be produced after the crossover and mutation. There are few cases in which the correction mechanism needs to be called. Please note that not all the constraints specified in the subsection 4.4.1 will be violated when applying the evolution operators.

1) **Violation of Utilization Constraint** (constraints 4.17 and 4.18) – in this case, the chromosome is modified by lowering the load of the node(s) with excessive utilization. The procedure randomly selects a runnable from one of these nodes and then moves it to a destination node, randomly selected among those that can accommodate the additional load. Runnables are moved until a feasible load distribution is found.
2) **Incorrect Definition of the Communication** (constraint 4.16) - if two communicating runnables are placed on different nodes, the gene in a chromosome that relates to the signal exchanged between the runnables must have a number associated with one of the buses that connect the two nodes. This correction mechanism checks all the values of genes related to signals. Each time an incorrect bus is found, the procedure randomly generates a new bus identifier among those that are valid with respect to the runnables allocation.

3) **Incorrect Partitioning and Ordering** (constraints 4.19, 4.20, 4.23, 4.24 and 4.25) – if the partitioning and ordering constraints are violated the correction mechanism will reassign the tasks/messages to runnables/signals. This is done by defining a set of possible partitioning configurations for runnables/signals and then choosing randomly one of them.

The constraint related to the observance of the deadlines assigned to transactions might also be violated. This is not handled by the correction mechanism as fixing such case would be too time consuming and would require some strategies to look for a feasible solution. In fact, the generation of an initial population itself might produce configurations violating this constraint. Nevertheless the evolution will lead to the populations with fewer chromosomes violating end-to-end deadlines and in a best case scenario, will lead to finding configuration characterized by the lowest response times, i.e. the optimal one.

4.4.4. Evaluation & Conclusions

The evaluation of this genetic approach has two goals. Assess quality of results and scalability. Assessment of the first is crucial for heuristic approaches such as the GA. This allows tuning the operators (crossover, mutation) in a way that will increase the effectiveness of the approach and see if a technique is capable of finding an optimal solution. The scalability on the other hand is a measure of the ability to handle large use-cases such as those present in the automotive domain.

The quality is assessed by comparing the solutions obtained with proposed technique to those which assure reachability of the optimum. The last can be an extensive search or technique such as the MILP (Mixed Integer Linear Programming). Naturally it would be the best to use only exact techniques for supporting the deployment, but as it will be shown, these are not scalable. Conclusions on scalability are made basing on the runtimes of the proposed GA technique and observations of how the quality decreases with the size of the input architecture.

Quality of Results – the optimal configurations for deployment as defined in this work for DD activation model are obtained using a MILP technique. The MILP formulation comes from [59]. A standard form of a MILP program is shown under eq. 4.40. The $x = (x_1, x_2, \dots, x_n)$ is a vector representing a solution to the problem. Search of this aims to maximize the objective function and needs to satisfy a set of constraints. There exist many solvers handling MILP formulations. The one used in this work is CPLEX. This technique will be used to solve other problems presented later in this thesis.

$$minimize\ c^T x$$
$$subject\ to\ Ax \leq b \qquad \textbf{4.40}$$
$$x \geq 0$$

First set of examples these are randomly generated small use-cases. Their specification shows the Figure 4.5 together with the hardware architecture on which the runnables will be deployed. In all these systems the WCETs of runnables and the WCTTs of signals are the same for all nodes and buses. The maximal utilization constraint is set to 1 for all the nodes and buses. There is one software component per runnable entity which means that there is no constraint which would bind the allocation of more than one runnable. This is not presented on the Figure 4.5 for the sake of simplicity. The next, Figure 4.6 presents the solutions obtained with our technique but also with the MILP. The optimization metric corresponding to these results is the maximization of the minimal slack (see eq. 4.13). The value of minimal slack is displayed for each found configuration from the Figure 4.6. The similarity of the obtained results for the two techniques proves a high quality of our approach when used for the small use-cases. When driving the deployment with the second metric, i.e. minimizing the sum of all the response times of the transactions (see eq. 4.12) proposed technique delivered the same, optimal results as MILP. The resulting configurations are the same for the use cases 1, 2, 3, 4 and 6. The solution for the use case nr 4 is different as shown on the Figure 4.7.

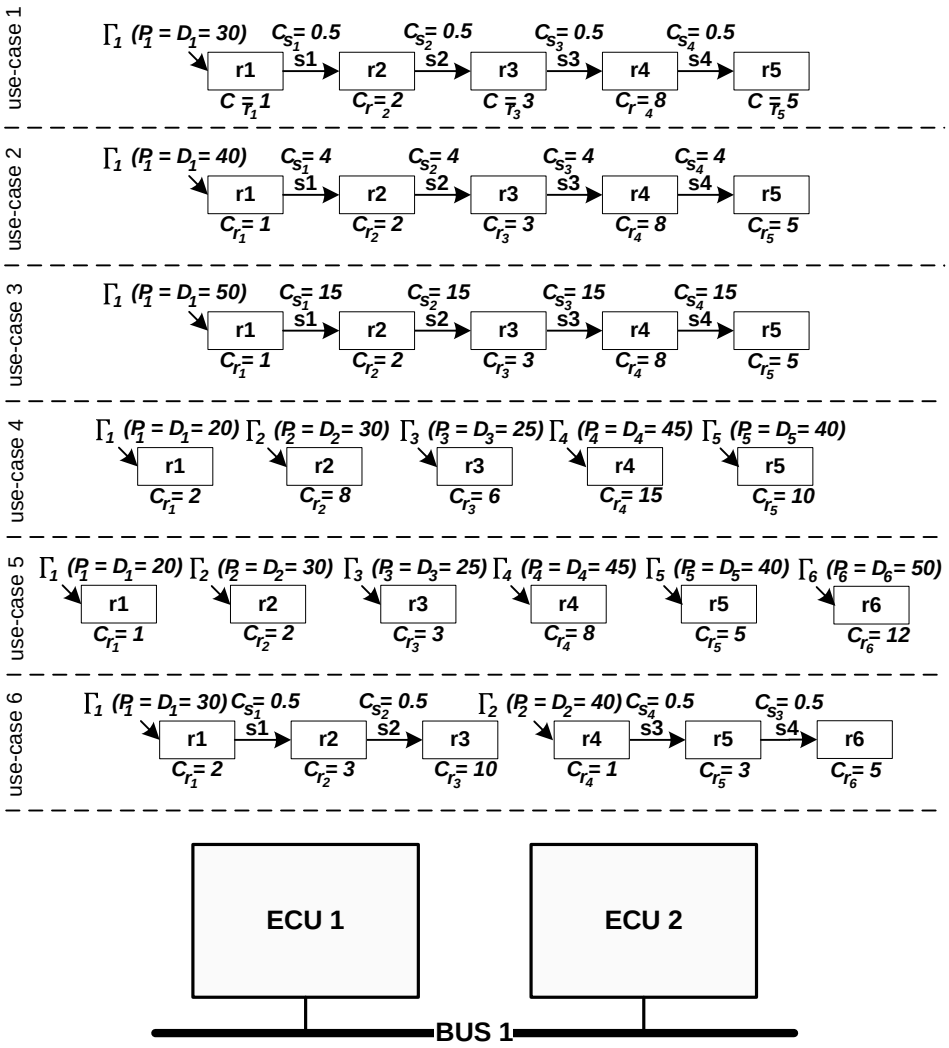

Figure 4.5. Simple Use-Cases

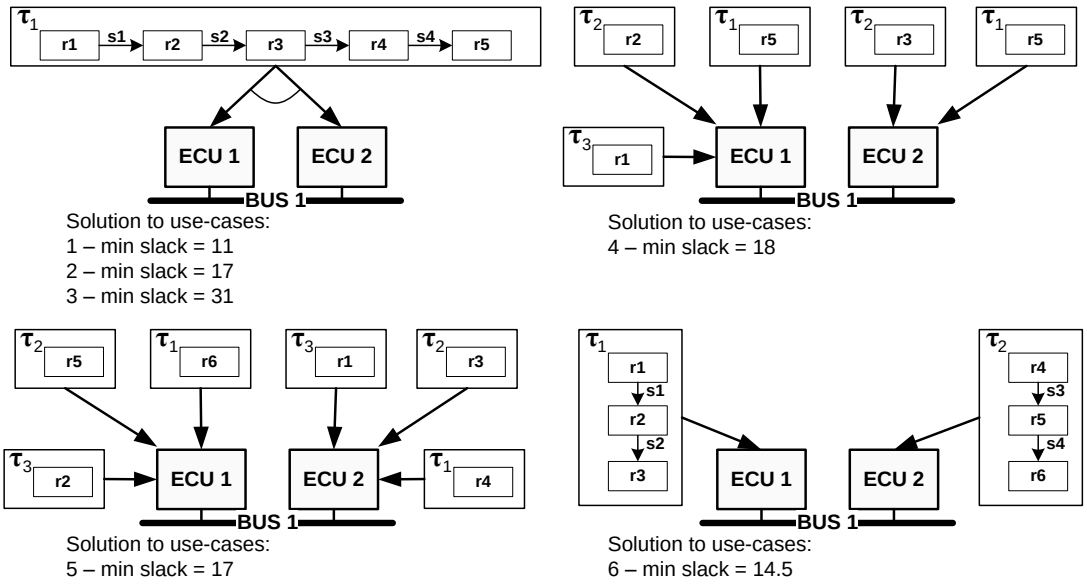

Figure 4.6. Solutions for the Simple Use-Cases obtained with the Metric 4.13

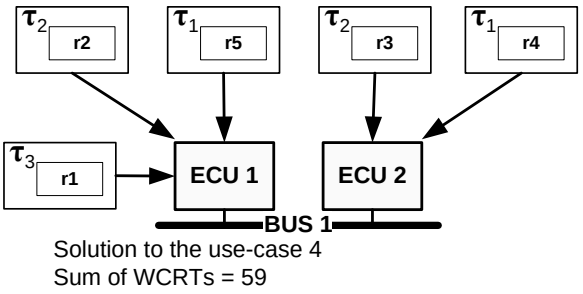

Figure 4.7. Solution for the Simple Use-Case nr 4 obtained with the Metric 4.12

These tests were run with an initial population GA_{pop} set to 1000. The criterion used for stopping the execution of the GA, i.e. GA_{stop} was that 20 consecutive evolutions bring the same result, i.e. the best configuration from the population doesn't change. Also when going to the next iteration of the proposed GA, the best configuration from the previous population will replace the worst configuration from the evolved population. Of course this happens only if the worst configuration from the new population is worse than the best from the old one.

Similar comparison between the MILP and GA techniques was done for a medium size use-case combining a Cruise Control System [60] and a Brake-by-Wire [61]. For both the obtained result was the same which implies that GA reached the optimum. The optimal configuration has

one transaction per one ECU, and for each transaction one task. The values for the minimal slack and latency are respectively 7.45 and 68.27.

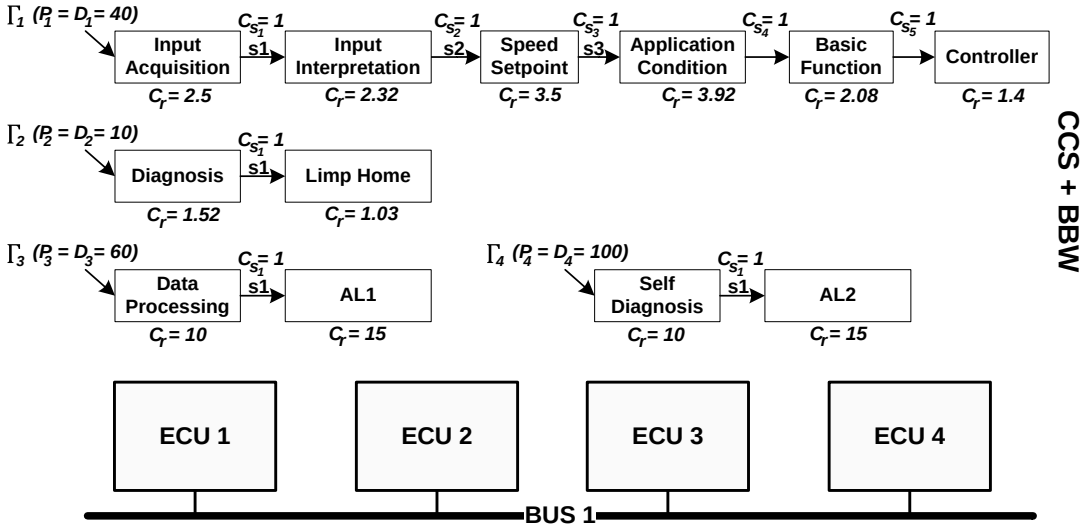

Figure 4.8. CCS + ABS System

In order to assess the quality for the larger use-cases a specific approach was used. This is due to the complexity of the problem to solve which turned out to be too complex even for the MILP technique and the CPLEX solver. Already for the use case with 40 runnables CPLEX returned with an out of memory error although run on a powerful machine, i.e. AMD Opteron™ 6164 HE processor (12 cores) running at 1.7 Ghz with 48GB of memory. In general CPLEX might finish computation with an error message not providing any result or it stops with an out of memory message. In the second case, it returns a result, however it is not sure whether it is optimal. Therefore specific use-cases were established for which optimal configurations (solutions) can be inferred. This has been done by first fixing a simple use-case shown in the Figure 4.9. For this use-case the optimal configuration contains only one task with all the runnables partitioned to it. Once the simple use-case has been fixed, other use-cases were found by replicating the simple use-case. This means that, each transaction, runnable, ECU and BUS is replicated. Hence when replicating by 11 the obtained use-case consists of 55 runnables, 11 ECUs and 11 BUSes. Also each ECU is connected to the original bus and all the replicas. For the replicated use-cases, the set of optimal configurations is characterized by having each ECU containing only one transaction (no inter-ECU communication). It is irrelevant for the transaction on which ECU it will be placed as the ECUs are homogenous due to the replication. This means that instead of

only one optimal configuration, their number equals to the number of combinations for the distribution of transactions on ECUs, assuming only one transaction per ECU.

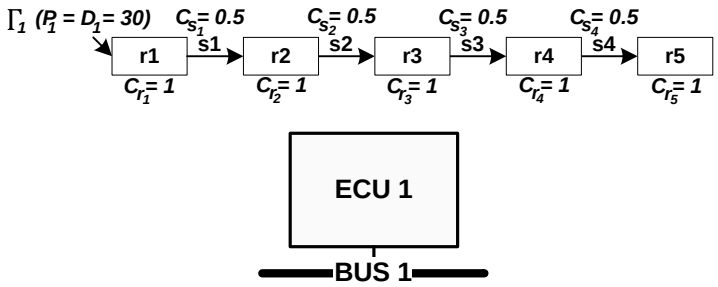

Figure 4.9. Non-replicated Use-Case

The first set of tests used the following set of parameters for the GA: $GA_{pop} = 1000$ and $GA_{stop} = 30$. The Figure 4.10 shows the difference between the optimal results and these obtained with the GA. Starting from the replication factor 6, i.e. 30 runnables and 6 ECUs, the obtained result was not optimal. Until the replication factor 11, starting from the 6th one, the sum of WCETs was on average 11,9% worse. The same use-cases were considered for the different configuration of the GA: $GA_{pop} = 50000$ and $GA_{stop} = 30$, i.e. with the bigger population. This gave an optimal result for all the replication factors but with the cost of an execution time. The two GA configurations were also launched for the replication factor 25, i.e. 125 runnables and 25 ECUs. The first GA configuration gave a fitness value 0,9486667 and the sum of WCETs equal 163,5. The optimal sum is 125. The second GA configuration returned fitness value 0,96666664 and the sum of WCETs 150.

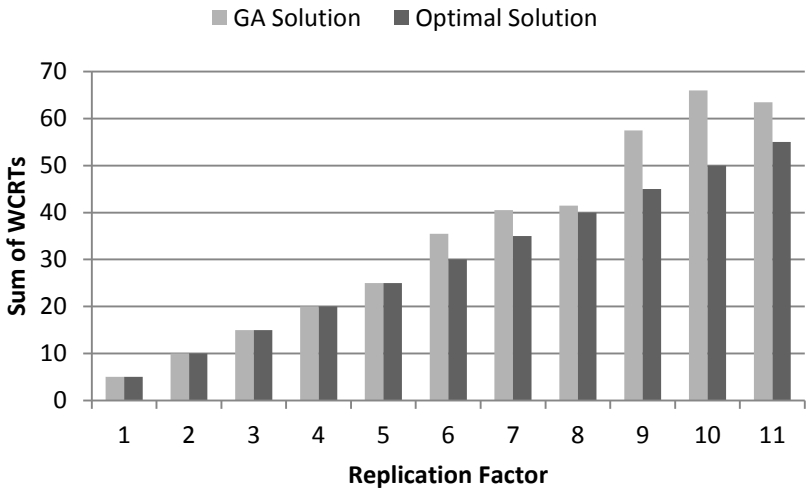

Figure 4.10. Comparison of the Optimal Solution with the Solution obtained with the GA.

Scalability – the Figure 4.11 compares the runtimes of the same GA with different number of initial population elements, i.e. 1000 and 50000. The use cases are the same as on the Figure 4.10. The runtime increases with the increase of the initial population. Nevertheless 2,82 hours of runtime for solving the use-case with replication factor 11, is acceptable. The runtimes for the replication factor 25 for the first and second GA configurations were correspondingly 0,88 hours and 19,72 hours. On an average the runtime of the second GA configuration is 42,7 times bigger. It is close to the expected value 50 as the initial population is 50 times bigger. The main reason why it is not perfectly close to 50 is due to the stopping condition which can lead to the different number of iterations. Therefore the average time was compared by proportionally computing the time it takes to run 100 iterations for each GA configuration. Then by comparing the times the runtime of the second configuration turned out to be 48,12 bigger than for the first GA configuration.

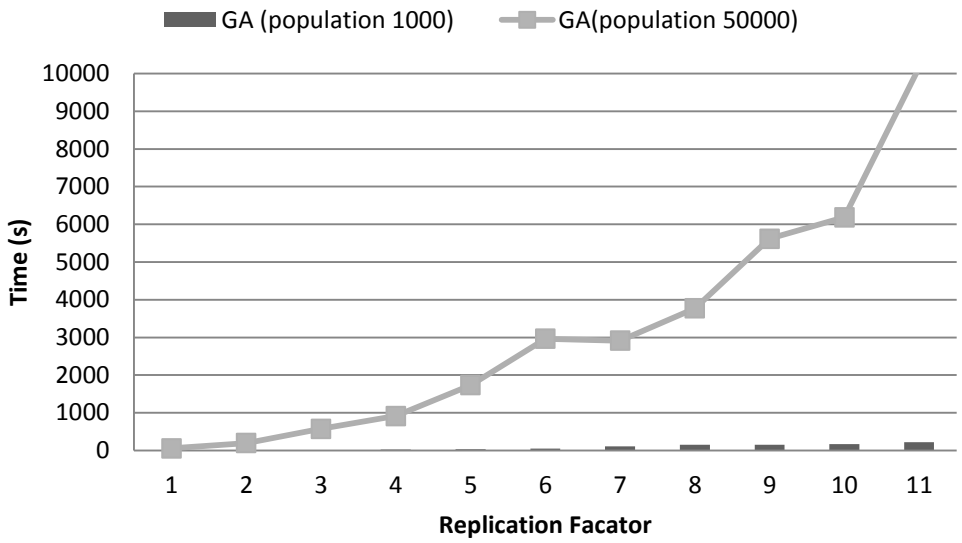

Figure 4.11. Runtimes for the GA with Different Initial Population

This part discussed adaptation of the GA technique to solve the deployment problem for the DD systems. It shows the applicability of the GA to holistically solve the deployment. Set of tests performed proves a high quality of obtained results. For those input architectures for which MILP was able to return solution with no error, GA was providing the same, optimal configuration. For larger use-cases the quality was increasing with an augmentation of the initial population size. This naturally causes the runtime increase but 2.82 hours can be easily considered as a small runtime to handle 55 runnables, 11 ECUs and BUSes. Even the runtime of 19.72 hours is acceptable (problem with 125 runnables, 25 ECUs and BUSes) as the benefit of providing a better configuration has a much higher value than the time spent to find it.

4.4.5. Technique for Optimized Deployment of TD

Technique for the deployment of TD similarly as in 4.4.3 is also based on the genetic algorithms. Although the nature of the problem is the same there are differences which influence the final specification of a genetic algorithm such as the encoding. The following description highlights these aspects of the GA for solving the deployment of the TD that differs from the previous specification.

Encoding – this encoding is also based on so called value encoding, in which each gene g_i (subset of bits) in a chromosome contains a specific value. Each gene relates either to a runnable

entity or a data signal. For the first, gene $g_i = ch_j(r_k)$ stores the value $V(g_i)$ representing runnable's allocation, partitioning and order, i.e. its index inside a task. For a data signal, value stored depends whether it is a global data signal or a data signal that is communicated locally. Value for a global data signal will hold information about the BUS and the message in which it is partitioned. If this is a local data signal s_i, value depends on whether s_i is communicated through the shared resource or no. For the first case, value represents one of the two mechanisms, either SL (value = 1) or RT (value = 2). For the second, value equals 0.

The gene value $V_{r_j}(g_i)$ for the runnable r_j is one number but stores information about the ECU number on which runnable r_j is allocated, the task number in which it is partitioned, and the position (order) inside the task. The $V_{r_j}(g_i)$ for runnable r_j for which selected ECU is e_k, task τ_l and position p is computed in a specific way, according to 4.41. The max_{ECU} is the maximal number of runnables that can be allocated on one ECU and max_{Task} is the maximal number of runnables that can be partitioned in one task. These values are automatically initialized before running the GA. The max_{ECU} is computed as a maximal number of runnables that can be hosted by one ECU without violation of utilization (for this WCETs and periods of runnables are used). The max_{Task} is computed based on the maximal number of runnables with harmonic periods.

$$V_{r_j}(g_i) = (k-1) * max_{ECU} * max_{Task} + (l-1) * max_{Task} + (p-1) \qquad \textbf{4.41}$$

The gene value for a data signal s_i, if transmitted on the bus, is computed in exactly the same way as in the case of the DD (see 4.39). If this is a local signal the gene value as mentioned before can be either 1 which relates to the SL (Semaphore Lock) used as a workaround to protect shared resource $\zeta'(s_i)$ or 0 to refer to the RT (Rate Transition block). Of course for some cases the shared resources and in consequence the specification of a protection mechanism is not necessary. The Figure 4.12 presents an example of a chromosome for a specific deployment configuration. For example the $V_{r_5}(g_8)$ means that the runnable r_5 is allocated on the ECU nr 2, partitioned on the task nr 2 with an index equal 1. Signal s_1 is a local signal that needs to be transmitted through the shared resource with specific protection mechanism. In this case $V_{s_1}(g_1) = 1$ which means that $\zeta'(s_1)$ is protected by the semaphore lock.

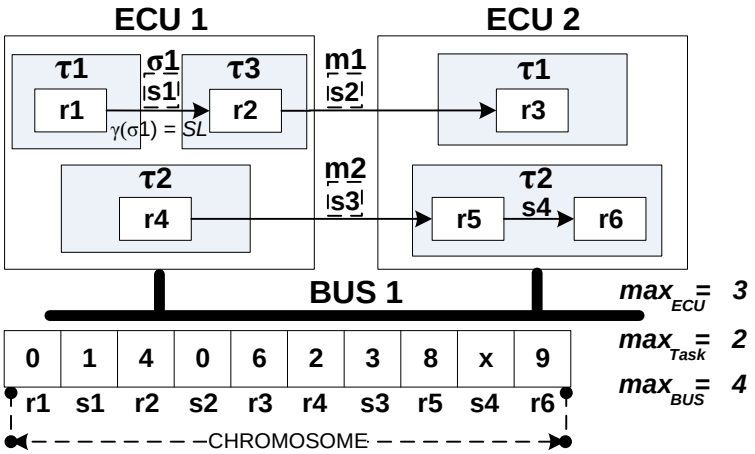

Figure 4.12. Example of a Chromosome for a Specific Deployment Configuration in the Context of the TD

Initial Population – the principle of generating the initial population is the same as in the 4.4.3. The generation produces correct chromosomes, i.e. they represent deployment configuration that respects the constraints formalized in the subsection 4.4.1.

Fitness Function – the fitness function used by this GA was specified in the subsection 4.4.1 under equation 4.32.

Evolution - the evolution of a population reuses the same mutation and crossover operators as in the context of the DD.

Correction Mechanism – the deployment of the TD extends the constraints specified for the DD. Therefore the correction mechanism for TD apart from accounting for the base constraints needs to also respond to the violation of those additional.

1) **Incorrect Indexes** (constraint 4.35) – if the assignment of an index for the runnable is not coherent with the constraint 4.35, the correction mechanism will first find the values of correct indexes for this runnable and then will randomly choose one of them.
2) **No Protection Mechanism for Shared Resource** (constraints 4.36 and 4.37) – all the signals that require the shared resource needs to be identified and for them a protection mechanism needs to be selected.

4.4.6. Evaluation & Conclusions

The evaluation has the same goal as in the case of the DD, i.e. assess the quality of the results and the scalability by measuring the runtimes. This subsection will also provide an evaluation

against approaches with none or partial partitioning. Their characteristic and occurrence in the current state of the art will be discussed later within this subsection. This comparison aims to show the improvement that can be gained in regards to the optimization metrics, if the partitioning is fully supported. All the tests will be run considering two configurations of the fitness function. First configuration focus only on the optimization of the end-to-end response times which means that the weight responsible for the memory optimization equals 0, i.e. $f(\Psi_i) = 1 * \mathrm{f}_{e2e}(\Psi_i) + 0 * \mathrm{f}_m(\Psi_i)$. In the second configuration the $f(\Psi_i) = 0.5 * \mathrm{f}_{e2e}(\Psi_i) + 0.5 * \mathrm{f}_m(\Psi_i)$.

Quality of Results – in order to assess the quality of results obtained with the GA, this technique was compared to the results obtained with the MILP. The problem of finding optimal configuration in the context of the TD is even more complex as for the DD due to the necessity for specifying the ordering. Consequently this evaluation uses similar approach serving to create use-cases for which the optimal configuration can be inferred. The initial use-case that will be replicated is that of presented on the Figure 4.13. The utilization constraint for the ECU 1 and its replicas equals 1. Each runnable is included in one software component. Please note that the replication assumes no changes for the characteristics of the ECUs/BUSes and therefore the WCETs/WCTTs of runnables/signals are the same on each ECU/BUS. For this non-replicated case, with the only optimization of response time, the set of optimal configurations contains any possible partitioning, and for each shared resource (if any) the protection mechanism is the RT. The left configuration presented in the Figure 4.14 is an example of an optimal solution for the simple use-case. For the replicated use-cases, the set of optimal configurations is characterized by having each ECU containing only one path (no inter-ECU communication). The right configuration from the Figure 4.14 shows an example of an optimal solution for the simple use-case if the fitness function accounts for the response-times and memory optimization. The entire path should be partitioned in one task. The replicated use-cases are optimally configured if each ECU contains only one path, partitioned in one task.

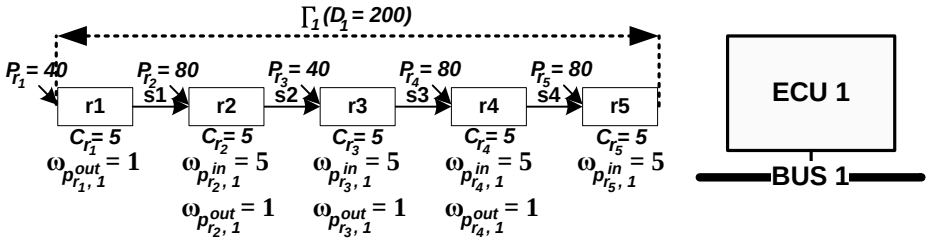

Figure 4.13. Non-replicated Use-Case with TD Semantics

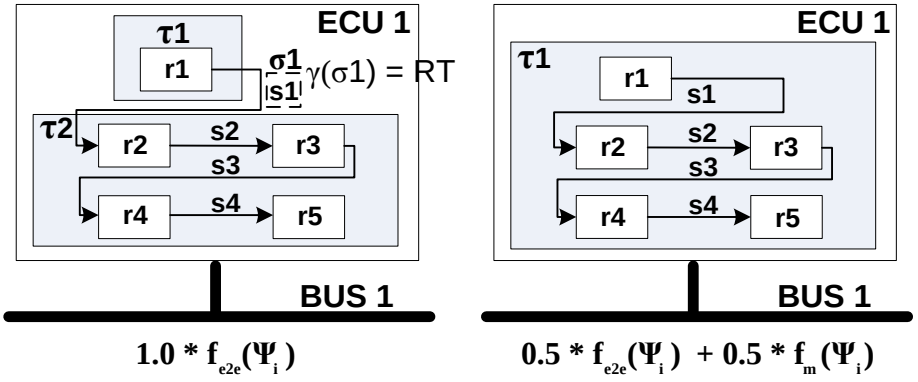

Figure 4.14. Optimal Configurations for Non-replicated Use-Case

The test results for the two metrics and two techniques (MILP and GA) are displayed on the two graphs Figure 4.15 and Figure 4.16. Indexes on the horizontal axes express the factor for the replication. As can be seen on the Figure 4.15, when architecture has been multiplied 6, 9, 10 and 11 times, the solver didn't return any solution. This was due to the returned error. For the factor 5, 7 and 8, the CPLEX finished execution with "out of memory exception". Nevertheless for the factor 5, returned result is optimal, which is not the case for 7 and 8. The GA for all the replication factors was able to return the optimal solution. The similar tests were run with weights 0.5 for the end-to-end responses and 0.5 for the memory optimization (see Figure 4.16). Already for the replication factor 5 the returned result was not optimal and starting from 9, CPLEX didn't provide any result. The degradation of the results given by the GA, started from factor 8. In general, the reason for this is that when using equal weights for the latency and memory optimization functions, the set of optimal configurations is smaller than if optimizing only end-to-end latencies. This is due to the fact that optimal solutions only have the runnables of the same path partitioned in the same task.

In all of those use-cases the GA was run with an initial population of 10000. The algorithm stops if during the 30 consecutive evolutions, the fittest chromosome doesn't change. When the GA was run on the population of 100000 it reached the optimal solution for the second metric and the replication from 8 to 11. This however increased the runtime to around 12 hours on a 2.4 GHz single processor computer with 8GB of memory.

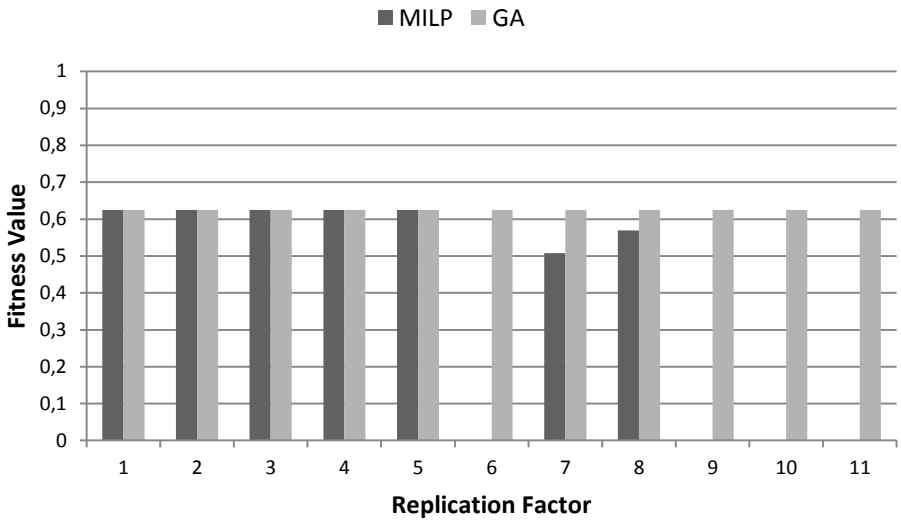

Figure 4.15. Results for MILP and GA ($1.0 * \mathbf{f}_{e2e}(\Psi_i)$)

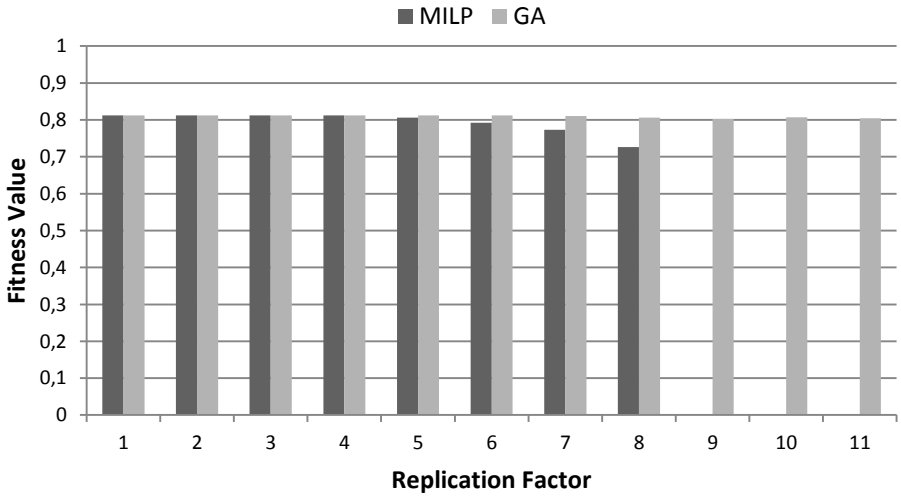

Figure 4.16. Results for MILP and GA ($0.5 * \mathbf{f}_{e2e}(\Psi_i) + 0.5 * \mathbf{f}_{m}(\Psi_i)$)

Scalability – the below charts display the runtimes needed to accomplish the previous tests. MILP on average gives the results in a shorter time. Concerning the Figure 4.17 the time was on average 41 times lower. This result would be much better if not the long execution for the use-case with the replication of 6 times. Running this use-case CPLEX experienced the problems with the memory and finally after the long run it returned with an error message. Similar problem occurred for the replication by 11 as is visible on the Figure 4.18. Although this decreased the advantage of the MILP runtime, it is still 36.7 times faster than the GA. Nevertheless the runtimes of the GA although much worse are still very acceptable, as finding the deployment with 55 runnables and 11 ECUs took only 1.01 hour when accounting only for timing responses and 1.4 hour when considering also the memory. Additionally the advantage of the GA runtimes is that they are predictable. Their increase was steady without any unexpected long runs as was the case for the MILP and the CPLEX solver.

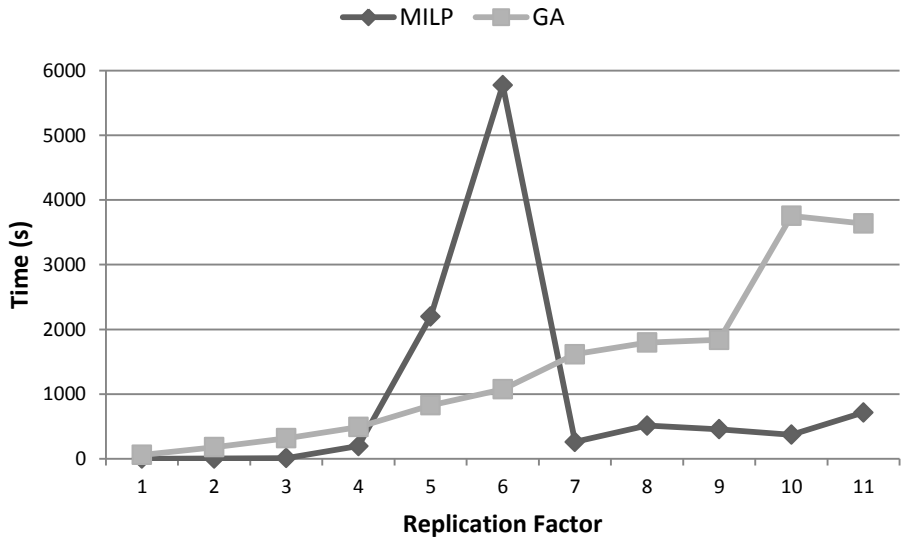

Figure 4.17. Runtime for MILP and GA ($1.0 * \mathbf{f}_{e2e}(\Psi_i)$)

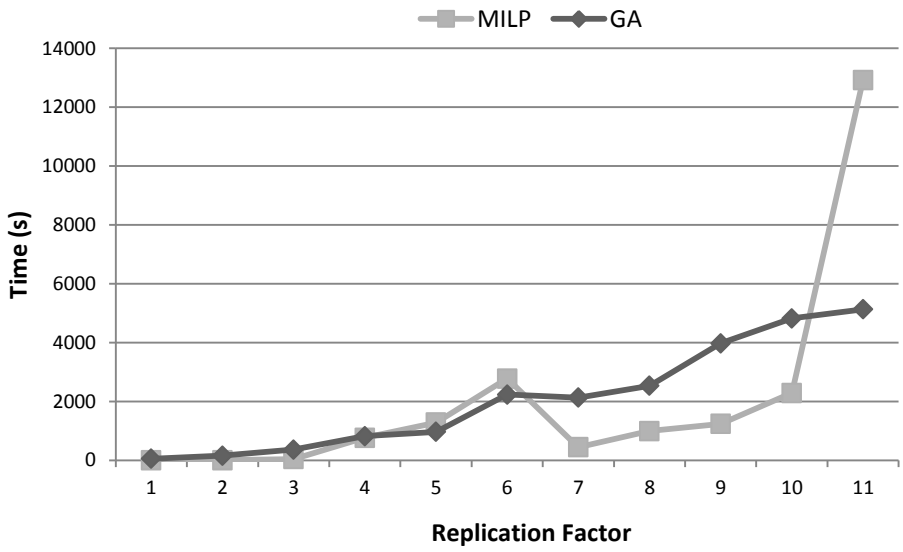

Figure 4.18. Runtime for MILP and GA ($0.5 * \mathbf{f}_{e2e}(\Psi_i) + 0.5 * \mathbf{f}_m(\Psi_i)$)

Evaluation Against Approaches with None or Partial Partitioning – this set of tests shows the added value of considering the partitioning. It will compare the results obtained with the GA with those that doesn't consider the partitioning (which is the case for [36]) or approaches that consider only partial partitioning, i.e. only runnables of the same period can be merged together (the case for [56]). The last two were implemented in MILP hence for them the obtained results are optimal in case when solver returned the result without error. The tests were run on a set of random input architectures.

Figure 4.19 and Figure 4.20 show that consideration of the partitioning has an impact on the optimization metrics. For the fitness 1.0 for the $\mathrm{f}_{e2e}(\Psi_i)$ (Figure 4.19) the GA obtained results 34.87% better than those with no partitioning and 16.48% from those with partial partitioning. For the last use-case, i.e. 35 runnables, 5 ECUs and 5 BUSes, MILP didn't provide any solution. Please note that for all of the approaches the same schedulability test is used (see subsection 4.2.2). Hence the metric improvement is a consequence of the constrained design space for the approach with partial and no partitioning.

Considering the Figure 4.20, the results of the GA were 6.8% better than those obtained with the approach disregarding partitioning, and 5.7% better from those which limit partitioning to the same periods. Let us note that for 35 runnables, CPLEX didn't return any result.

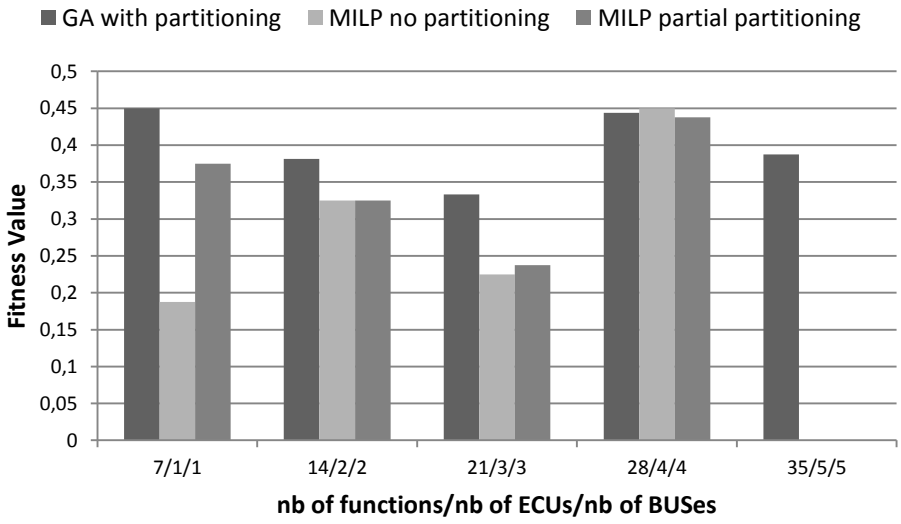

Figure 4.19. Comparison using Fitness Function ($1.0 * \mathbf{f}_{e2e}(\Psi_i)$)

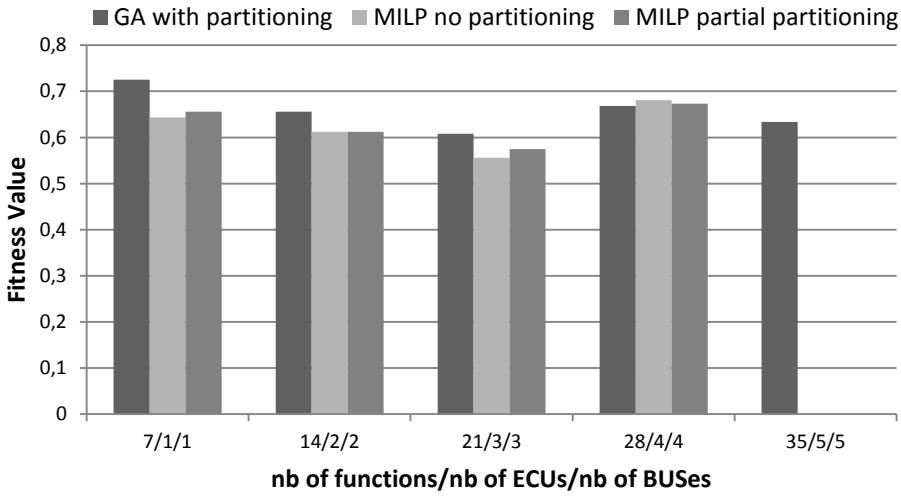

Figure 4.20. Comparison using Fitness Function ($0.5 * \mathbf{f}_{e2e}(\Psi_i) + 0.5 * \mathbf{f}_m(\Psi_i)$)

Similarly as in the case of the DD, genetic algorithms are applicable for solving the deployment also in the context of the TD systems. In addition to the standard evaluation, i.e. the quality and scalability, several tests were run to compare with the state-of-the art techniques. The results show the significant improvement of the system timing and memory consumption if the partitioning is handled by the optimization technique. The last is not the case for the current approaches.

The two proposed techniques (subsections 4.4.3and 4.4.5) which try to solve the deployment problem as defined for the DD and TD in one step are very efficient for the medium-sized input architectures, i.e. those that contain around 50 runnables. It is however desirable to be able to tackle problems of larger sizes, i.e. with 200 runnables and more. This motivated the work on improving the holistic approach. Result in a form of so called Two-Step Approach is described in the next section.

4.5. Two-Step Approach

It was shown that the quality of the results given by the GA can be improved by increasing the population size. This however leads to the increase of the algorithm runtime. In order to improve the results without boosting the runtime this section presents a two-step approach called in this work TSDA (Two-Step Deployment Approach). The TSDA is a heuristic which fundamental part is a combination of two strategies: divide-and-conquer (DaD) and iterative improvement.

The DaD strategy serves in this case to organize the deployment into two sub-problems. These sub-problems differ depending on the activation model. For the DD the two steps are: **(1)** allocation of tasks/messages on ECUs/BUSes and **(2)** partitioning and scheduling of runnables/signals in tasks/messages considering the allocation from the previous step. The Figure 4.21 describes the principle of the TSDA. Please note that the first step, i.e. the allocation is done for tasks and messages. This means that the tasks and messages are defined, i.e. the partitioning of runnables/signals is known as well as priorities assignment. Therefore the tasks/messages allocation will determine the runnables/signals allocation which then can be repartitioned in the second step to improve the configuration. The solution for the partitioning and scheduling comes either from the Step 2 and is delivered with the inner loop or in the case of the initial run of the algorithm, it comes as an "initial configuration". The last can be provided by the designer or with some predefined strategies, e.g. one runnable/signal to one task/message or randomly generated. The evaluation subsection 4.5.3 will study the influence of the initial configuration (IC) on the final result.

The iterative improvement guides the solution towards the optimum. This strategy is implemented at two levels of the proposed algorithm; inner and outer loops. The inner loop tries to find an optimal system configuration by applying iteratively an optimization sequence until convergence which is reached if two consecutive inner iterations deliver the same result. As it

can be seen on the Figure 4.21 the inner loop embeds two phases. These are (1) optimization of the tasks/signals allocation (Step 1) and optimization of the partitioning and scheduling of a given allocation (Step 2). The final result of an inner loop might represent a local optimum. In order to escape from the local optimum, the outer loop is used. This requires providing a new initial configuration.

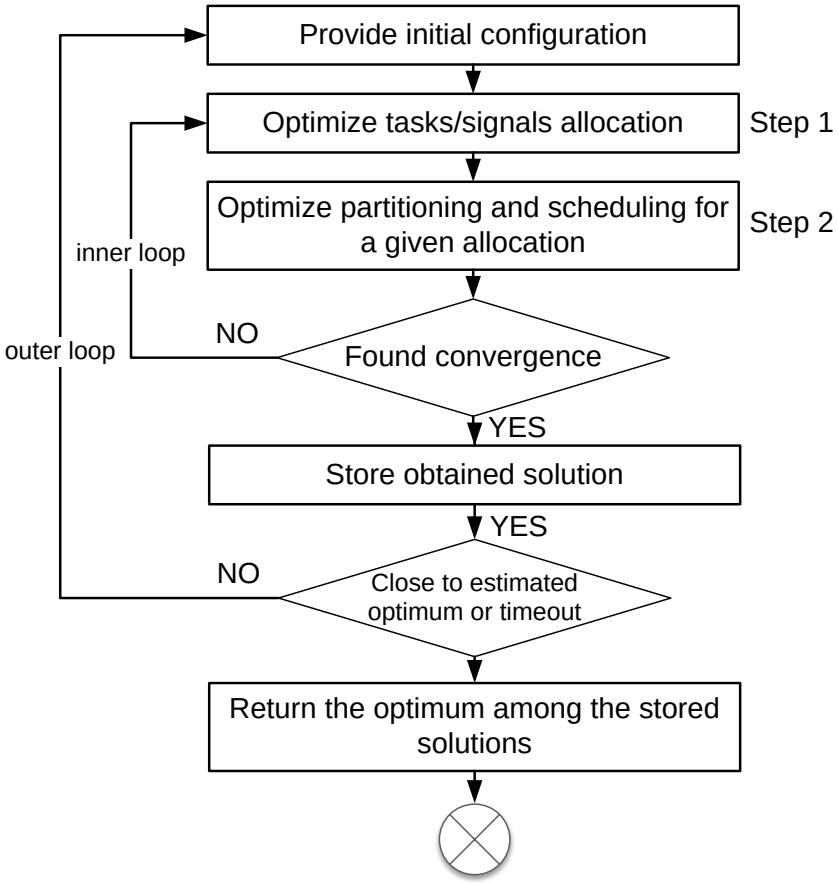

Figure 4.21. The Two-Steps Deployment Approach (TSDA)

4.5.1. GA Formulation for the Two-Step Approach

This subsection provides the specification of the genetic algorithms used to solve the two main steps of the TSDA approach, i.e. Step 1 and Step 2. All the other phases of the TSDA are intuitive. The "Provide initial configuration" was discussed before. Test for checking the convergence ("Found convergence") simply compares the fitness values of the previous and the current solution obtained within the inner loop. If convergence is reached, the solution will simply be remembered so later it can be compared at the level of the outer loop (block "Store obtained solution"). The decision block "Close to estimated optimum or timeout" is a simple

condition that will check if the solution is satisfactory in regards to the optimization metric or whether the overall runtime of the approach reach the predefined timeout. The last block simply returns the best configuration from among all the stored solutions.

A GA Solution to the Allocation Problem (Step 1)

Encoding – in the allocation problem, a specific solution, i.e. a single chromosome ch_i, represents the allocation of tasks onto the processing units, and messages on buses. Each gene relates either to a task or a message. For the first, gene $g_i = ch_j(\tau_k)$ stores the value $V(g_i)$ representing task's allocation. For a message m_k its corresponding gene $g_i = ch_j(m_k)$ holds value of the allocation BUS. The Figure 4.22 presents an exemplary chromosome for a specific tasks/messages allocation.

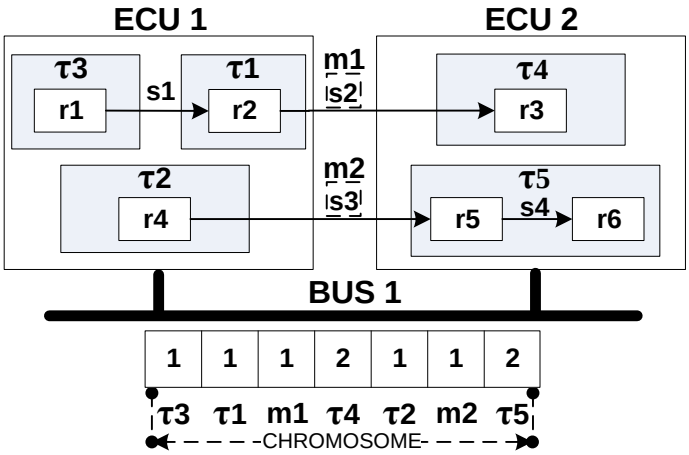

Figure 4.22. Example of a Chromosome for a particular Allocation Configuration

Initial Population - population is generated randomly, i.e. for each task gene, a random number representing its execution node is assigned. However, the initial population does not contain solutions which violate the utilization constraints. Therefore if a generated chromosome leads to the violation of a utilization constraint, a correction procedure is being called.

Fitness Function – the fitness function is the same as specified for the technique for the holistic deployment of the DD (see 4.4.1).

Evolution – the evolution operators are the same, i.e. OX3 crossover operator and mutation operator that randomly choice a gene to change its value (considering only the correct ones during the change).

Correction Mechanism - in the case of the violation of utilization constraint, the chromosome is modified by lowering the load of the node(s) with excessive utilization. The procedure randomly selects a task from one of these nodes and then moves it to a destination node, randomly selected among those that can accommodate the additional load. Tasks are moved until a feasible load distribution is found. Incorrect definitions of the communication are also fixed. If two communicating tasks are placed on different nodes, the gene in a chromosome that relates to the message exchanged between the tasks must have a number associated with one of the buses that connect the two nodes. Our correction mechanism checks all message values. Each time an incorrect bus is found, the procedure randomly generates a new bus identifier among those that are valid with respect to the tasks placement.

The GA Formulation for the Partitioning and Scheduling (Step 2)

After the definition of the runnables and signals allocation (implicitly by the placement of tasks and messages) the maximum number of new possible tasks and messages for each node and bus can be computed as the number of runnables or signals allocated on the resource. Also, signals that result in local communications are not represented in chromosomes. For the second step, only the encoding, the generation of the initial population and the correction mechanism are described. The crossover mutation operators follow the same logic as in the allocation stage. The fitness function is the same.

Encoding - each gene represents a runnable or a signal exchanged among CPUs. The value of the gene is the index of the task or message executing the runnable or transmitting the signal. The index of a task or message also represents its priority, and its period is the gcd of the runnables/signals mapped onto it. In the case of the configuration from the Figure 4.22, the system partitioning and scheduling is represented by the chromosome shown in the Figure 4.23, where r_1 is executed by τ_1 (with priority 1).

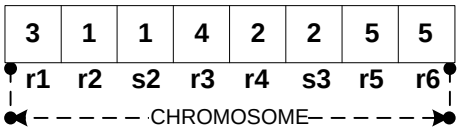

Figure 4.23. Example chromosome for the Partitioning and Scheduling Configuration

Correction Mechanism - called when a new chromosome is generated as part of the initial population or after the crossover and mutation enforces the order of execution constraints. The range of values for a gene is constrained by the values assigned to other genes. If runnable r_i precedes r_j, and the gene representing r_i is assigned to a task with priority π_{τ_k}, then r_j should be partitioned on the same task or a task with priority lower than π_{τ_k}.

4.5.2. Establishment of the Global Order

The Step 1 takes as an input configuration which is either the "initial configuration" or the configuration obtained from running the Step 2. The problem that occurs is that the priorities order between the tasks and messages delivered with the input configuration is local (local order). This means that the priorities relation holds valid within the boundaries of a single ECU/BUS (note that the input configuration provides information about the specific allocation). Valid means that it obeys the constraints as specified in 4.4.1. However in the global context, i.e. when looking for a new allocation for given tasks/messages, the local order might no longer be valid for all the allocation configurations. Therefore it is necessary to establish a global order out of the local, i.e. the one that will be valid for all tasks/messages allocation combinations. To ease the understanding let's consider an example input configuration as visible on the Figure 4.24 and assume that the task/message index represents its priority and the higher the index the higher is the priority. Such input configuration can result from the GA run. The Step 1 will take from this configuration only information about the partitioning and scheduling and will look for an optimal allocation (of course it might happen that the allocation from this figure is in fact the optimal one). Considering such partitioning and scheduling certain, possible allocation configurations might be invalid in regards to the local total order and runnables order constraints. For example a configuration in which τ_1 is allocated on ECU 2 will violate the local total order constraint because r_2 should be partitioning in task with a priority higher than the priority of a task hosting r_3 or partitioned on the same task. Therefore the technique developed for the Step 1 wouldn't allow such configuration. This is however not the best solution because it shrinks the space of

possible configurations to consider. A workaround is to establish priorities order called in this work, the global order. The main feature of a global order in regards to the local order is that for all the configurations in which local order holds valid, global order is also valid and gives the same results for the schedulability analysis test. The global order for the example from the Figure 4.24 is shown on the Figure 4.25. Please note that on this figure the priorities are not represented with the task indexes but with an additional notation π_{τ_i}/π_{m_i}.

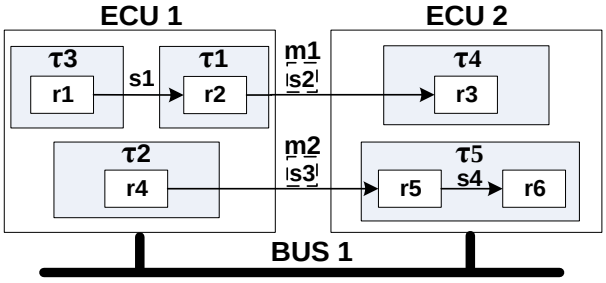

Figure 4.24. Example of the Input Configuration

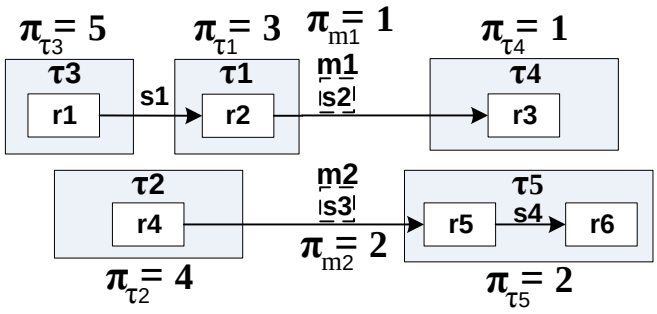

Figure 4.25. Global Order for the Example of the Figure 4.24

The following is a presentation of a MILP formulation (parameters, constraints and objective function) used to establish a global order for an input configuration of Step 1.

Parameters: the parameters are the tasks' and messages' priorities π_{τ_i}/π_{m_i}.

Constraints:

1) Local order constraints – the global order should respect the properties of the local order, i.e. the priorities order between the tasks/messages belonging to the same ECU/BUS.
2) Global runnables order – this constraint is similar to the runnables order constraint specified in 4.4.1. The last is specified in the context of an ECU/BUS whereas in this case the context is global.

$$\bigwedge_{r_i, r_j \in R} \Gamma(r_i) = \Gamma(r_j) \wedge i < j \Rightarrow \pi_{\tau(r_i)} \geq \pi_{\tau(r_j)} \quad \textbf{4.42}$$

3) Global signals order – this constraint is analogous to the above "Global runnables order constraint".

$$\bigwedge_{s_i, s_j \in S} \Gamma(s_i) = \Gamma(s_j) \wedge i < j \Rightarrow \pi_{m(s_i)} \geq \pi_{m(s_j)} \quad \textbf{4.43}$$

4) Minimum & maximum constraint – this constraint specifies the minimal and maximal allowed value for the task/message priority.

$$\bigwedge_{\pi_{\tau_i}} 1 \leq \pi_{\tau_i} \leq |T| \quad \textbf{4.44}$$

$$\bigwedge_{\pi_{m_i}} 1 \leq \pi_{m_i} \leq |M| \quad \textbf{4.45}$$

Objective Function: there is no optimization objective driving the search of the global order. The only goal is to find priorities that will respect all the above constraints.

4.5.3. Evaluation & Conclusions

The evaluation has the following objectives: (1) compare the quality of results obtained with the TSDA and the holistic approach, (2) compare the runtimes of the TSDA and the holistic approach and (3) see how the different initial configurations influence the final result.

The evaluation of the results quality is based on the same use-case as defined in 4.4.4 under the Figure 4.9 and its replications. It was shown that for them the optimal solution is known. In this set of tests the initial configuration was very simple, i.e. one task per one runnable entity. The Figure 4.26 shows an initial configuration for the simple use-case where the index of the task represents also its priority. The higher the value the higher is the priority. The rule for the initial configuration of the replicated use case is that the runnables' replicas will have the task with a priority smaller than that of the original runnable by the factor 5**replica_index*. For instance for the replication factor 2, all the replicated runnables have the replica index equal 1, i.e. runnable r_i where $i \in \{1, 2, 3, 4, 5\}$ will have only one replica, $r_{i,1}$. Runnable r_1 will be partitioned in task

with priority 10 and its replica, $r_{1,1}$ on the task with priority 5. Similarly each signal is partitioned in one message and the priority of the message equals the priority of its sending task.

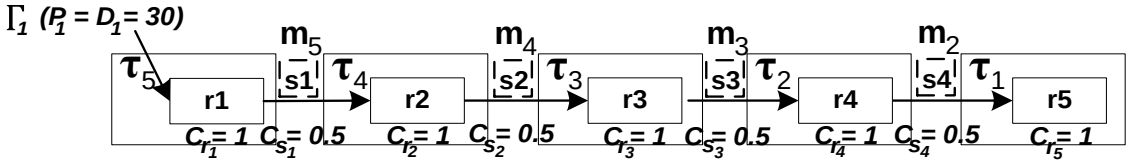

Figure 4.26. Initial Configuration for the Simple Use-Case

The comparison of the results obtained with the One Step GA (OS-GA) and the GA Two Steps Approaches (TSDA-GA) in regards to the optimal solution is shown on the Figure 4.27. Both were run with the initial population of 1000 chromosomes, i.e. OS-GA and the Steps 1 and 2 of the TSDA-GA. The GA One Step was already evaluated in the context of these use-cases and it was shown that starting from the replication factor 6 it didn't provide an optimal result. On average the sums of WCRTs were 11.9% worse when considering only those input architectures for which the provided solution was not optimal. The TSDA-GA didn't give an optimal solution already for the factor 5, however for the 7 the optimal solution was reached which wasn't the case for the OS-GA. These results were 10,62% worse from the optimal result, accounting for those use-cases for which result was not optimal. This shows that in this set of tests the TSDA-GA performed better. The TSDA-GA was also tested for the replication factor 25 and the obtained sum of WCRTs was 136,5 which is only 9,2% worse than the optimal solution. This result is much better from the OS-GA which was 163,5 when run on the same population. In fact the result was still better from the OS-GA when the last was run on the population with 50000 chromosomes. The sum of WCRTs that it returned was 150. For this size of a population the TSDA-GA improved the result and returned 133. The TSDA-GA was also tested with the replication 50 which gives us 250 runnables, 50 ECUs and 50 BUSes. The optimal sum of WCRTs is 250 and the obtained result was 281 which is 12,4% worse.

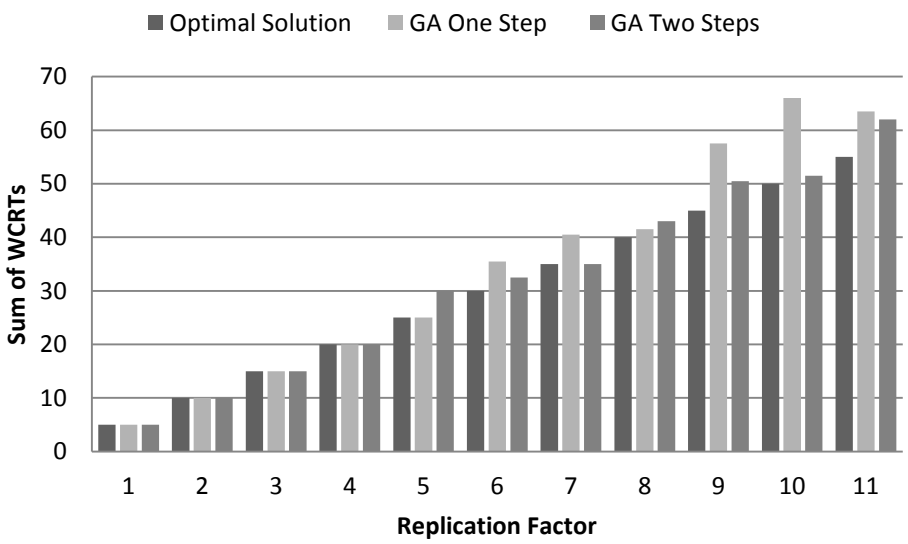

Figure 4.27. Comparison of the Two Steps Approach with the Holistic Approach and the Optimal Solution

The significant feature of the TSDA-GA is that it provides good results for larger use-cases in a time that is comparable to the runtime of the OS-GA or even better. For small use-cases its runtime is worse. This is due to the fact that although the optimal result might be found in the first iteration of the inner loop, the second iteration needs to be run to check whether the algorithm converges. This overhead is negligible for the larger use-cases. Therefore as it can be seen starting from the replication factor 7, the runtimes of the TSDA-GA were better, with only exception for the replication factor 11, where the runtime of the OS-GA was only slightly better. Concerning the replications by 25 and 50 the runtimes were correspondingly 1809,623s (0,5h) and 7352,271s (2,04h) which is highly acceptable. The OS-GA with replication 25 returned result after 0.9h which is almost twice longer. These results were for the initial population of 1000. When run with 50000 initial chromosomes the runtime of the TSDA-GA for the replication 25 was 41102,221s (11,42h) and for the OS-GA 70980,911s (19,72h).

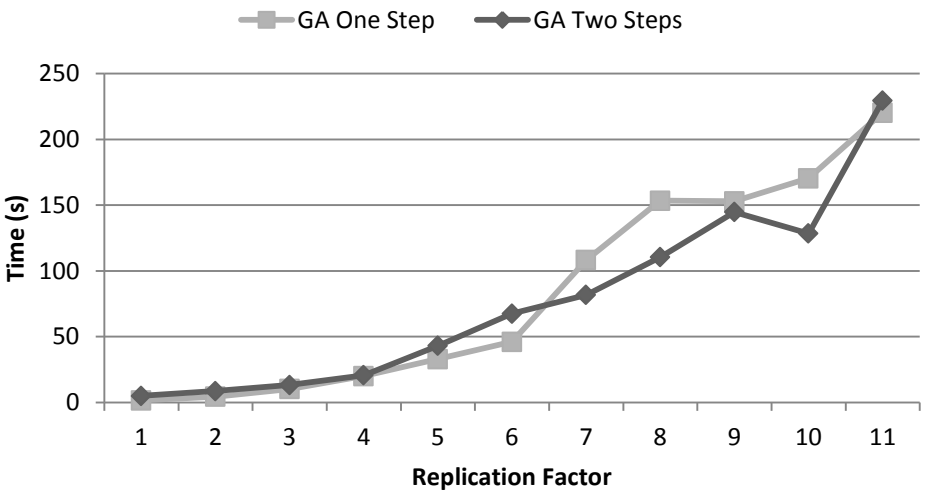

Figure 4.28. Runtimes of the OS-GA and the TSDA-GA

The above tests were run with the specific initial configuration assuming one task per one runnable and analogously for signals, each of them was partitioned in a separate message. There might be multiple other configurations to consider. Their number equals to the amount of combinations for the correct partitioning and scheduling respecting the constraints from the 4.4.1 (both for the DD or TD) plus the constraints specified for the global order. The choice of the initial configuration highly impacts the final result but also the number of the iterations until convergence which has an implicit effect on the runtime. The following tests are performed to show this. The chosen use-case is the one from the Figure 4.8 which is a combination of the Cruise Control System and the Anti-lock Braking System. The considered initial configurations are shown on the Figure 4.29. This figure discards the information about the periods, deadlines, WCETs, etc. This timing information was already included in the Figure 4.8.

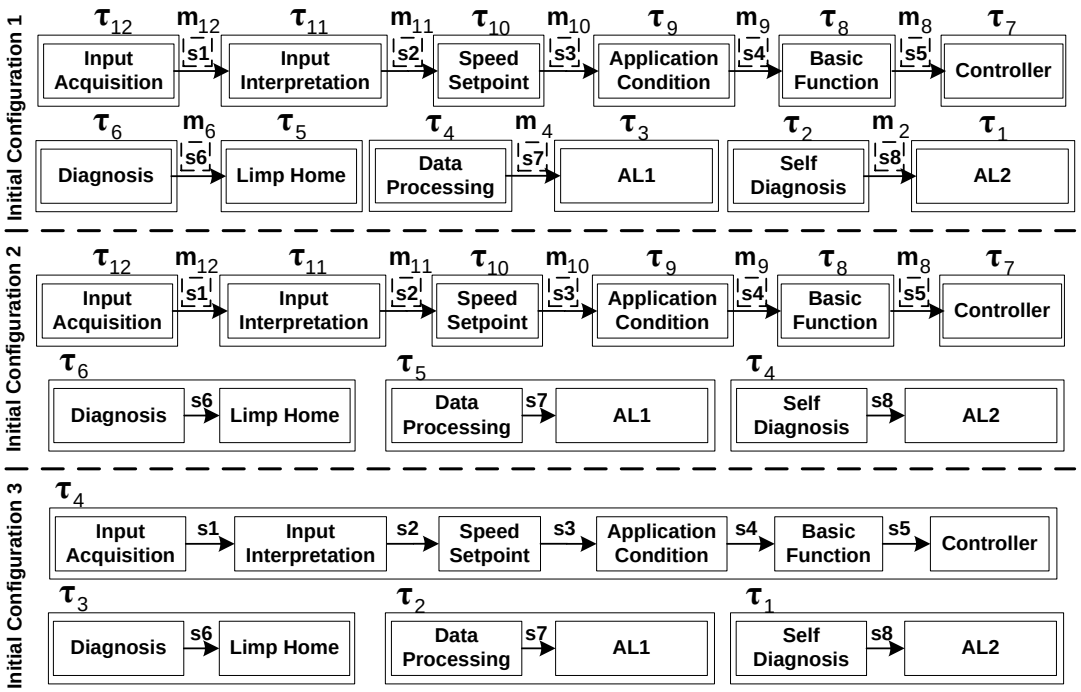

Figure 4.29. Initial Configurations for the ABS + CCS

When running the OS-GA on this use-case the obtained sum of WCRTs was 68,27. This is the optimal solution which was proven by obtaining the same result using the MILP formulation from [59]. The table below (Table 4.7) shows the results obtained with the TSDA-GA considering different initial configurations. This table also shows the intermediate results obtained in each iteration and also in each step, i.e. allocation (Step 1) and scheduling & partitioning (Step 2). The first and second initial configurations didn't lead to the optimal solution. The strategy employed in the initial configuration 1 is the same as in the IC used in the previous tests. This strategy delivered optimal results for the replication factors 1 to 5 and 7 but in this example it proves not to be the best choice. Importantly, the intermediate results were always optimal (i.e. results obtained for the Step 1 and 2). This knowledge is due to the comparison to what has been obtained with the MILP formulation present in [59]. The last was specified for each step, i.e. Step 1 and 2. This confirms that the problem lies not in the technique implementing consecutive steps but in the initial configuration itself. The same result is delivered when using the IC 2. The optimal solution was reached for the third initial configuration. Worth noting is also the number of iterations that differs among the initial configurations. The IC 3 hasn't only lead to the best solution but reached it in two iterations. In fact this result was

obtained after one iteration but in order to evaluate the convergence, the second iteration had to be run as well.

		Iteration				
Initial Config.	**Step**	1	2	3	4	5
1	1	112,69	96,67	78,8	72,75	71,82
	2	103,09	81,85	76,23	71,82	71,82
2	1	89,47	74,32	71,82	-	-
	2	78,32	71,82	71,82	-	-
3	1	68,27	68,27	-	-	-
	2	68,27	68,27	-	-	-

Table 4.7. Intermediate Results for each Initial Configuration

The TSDA is a workaround to the scalability issue that poses a big challenge for the deployment strategies. One way to deal with it is the usage of the heuristic approaches such as the genetic algorithms which outperform the MILP in a holistic approach. They serve well for the medium-sized problems for which they are capable of finding an optimal deployment configuration. The TSDA reveals its usefulness for large input architectures. Its drawback lies in the necessity of providing an initial configuration which has an impact on the final result. It might happen that for small or medium-sized use-cases wrong choice of the IC will result in a solution that is not optimal. Therefore it is advisable to use the TSDA rather for larger problems as for them the holistic approach might be less efficient. This was shown by considering the input architectures with 125 and 250 runnables for which the TSDA was able to provide a result close to the optimal and better when compared to the holistic approach.

4.6. Evaluation of the new Methodology

The change done in the EAST-ADL2/AUTOSAR methodology as described in the section 4.3 suggests postponing the decision about the allocation of the functional entities (atomic functions at the EAST-ADL2 level, runnable entities at the AUTOSAR level) until the implementation level, i.e. the AUTOSAR. This enables holistic consideration of a deployment problem at the AUTOSAR level as the allocation will not be fixed within the EAST-ADL2 model. In order to show the benefit of the holistic approach, this section compares described

techniques with the approach compliant with the current status of the EAST-ADL2/AUTOSAR methodology. The comparison is done within the context of the DD activation model. For this purpose a two staged technique was designed (don't confuse with the Two-Step Approach presented in 4.5) that will be called Methodology Compliant Deployment Technique (MCDT). This technique first supports the allocation of the atomic functions done at the EAST-ADL2 level. Then, as these functions will be transformed into the runnable entities, the last will already have a fixed allocation. Therefore the next part of the technique supports the partitioning and scheduling. In the following paragraphs two stages of the technique compliant with the EAST-ADL2/AUTOSAR methodology are described.

4.6.1. Allocation at the EAST-ADL2 Level

The allocation of the atomic functions (represented with af_j, and its hosting ECU with the $E(af_j)$) on the ECUs and exchanged signals on the BUSes is supported by the GA implementation. The mutation and crossover operators are similar as in all the previous implementations. The encoding can be compared to this described in 4.5.1 but the gene in this case represents either atomic function or exchanged signal, not task and message. Values stored correspond to the number of the ECU/BUS. The optimization objective in this case is the minimization of the utilization of each ECU/BUS (see eq. 4.46) as defined in [62]. Observance of the utilization constraint is the necessary condition for the system to be schedulable. The timing metric such as this used by the holistic approaches cannot be computed here as there is no information about the OS tasks and messages specification of which is necessary to run the schedulability test. This is the main disadvantage of the EAST-ADL2/AUTOSAR methodology. Namely the final concern, i.e. the minimization of the end-to-end responses cannot be considered throughout the entire development process. The optimization of utilization is a way to abstract the final metric at the higher level, however it doesn't necessarily lead to the optimal response times when configuring architecture at the AUTSOAR level with fixed allocation. Nevertheless it is the only timing metric that can be used at this level considering the input information. Additional clarification is necessary for the full understanding of the eq. 4.46. The $U(e_i)$ is the utilization computed for the ECU e_i, i.e. $U(e_i) = \sum_{E(af_j)=e_i} \frac{C_{r_j}}{P_{r_j}}$. Similarly $U(b_i)$ represents the utilization computed for the bus b_i; $U(b_i) = \sum_{B(s_j)=b_i} \frac{C_{s_j}}{P_{s_j}}$.

$$f_U = \begin{cases} 1 - \dfrac{\sum_{e_i} \dfrac{U(e_i)}{U_{e_i}^{ecu}} + \sum_{b_j} \dfrac{U(b_j)}{U_{b_j}^{bus}}}{|E| + |\beta|}, & \forall_{e_i} U(e_i) \le U_{e_i}^{ecu} \wedge \forall_{b_j} U(b_j) \le U_{b_j}^{bus} \\ 1 - \dfrac{\sum_{e_i \in \{e_k : U(e_k) > uc_{e_k}\}} \dfrac{U(e_i)}{U_{e_i}^{ecu}} + \sum_{b_j \in \{b_k : U(b_k) > U_{b_k}^{bus}\}} \dfrac{U(b_j)}{U_{b_j}^{bus}}}{\left|\{e_i : U(e_i) > U_{e_i}^{ecu}\} \cup \{b_j : U(b_j) > U_{b_j}^{bus}\}\right|} \end{cases} \quad \textbf{4.46}$$

Apart from the utilization metric objectives of other types can be considered at this level. Examples are the cost or safety [62]. The cost can be expressed by the number of hardware resources used, i.e. ECUs and BUSes. The functional safety corresponds to the number of replicas provided for the safety critical functions. Redundancy is regarded as a safety architecture concept according to the safety standard ISO 26262 [28]. Nevertheless as mainly the timing characteristics will be compared when evaluating the new, refined methodology the presented technique accounts only for the utilization metric.

4.6.2. Partitioning and Scheduling at the AUTOSAR level

The partitioning and scheduling are done at the AUTOSAR level on the runnables and data signals. This activity is also supported by the two staged technique. In fact this part reuses the GA implementation as defined in the 4.5.1 under "The GA Formulation for the Partitioning and Scheduling". At this stage schedulability test can be run. Consequently the metric used at this level refers to the end-to-end responses, i.e. the final objective to be optimized.

4.6.3. Evaluation & Conclusions

The goal of this subsection is to compare the results obtained by running the MCDT and the holistic approach, compliant with the refined EAST-ADL2/AUTOSAR methodology. The selected holistic approach is the one from the subsection 4.4.3. The input architectures for comparison are the following:

- Simple use-cases from the Figure 4.5
- CCS+ABS from the Figure 4.8
- Use-case for replication with replication factors from 2 to 6 (Figure 4.9)

The results as presented on the Figure 4.30 clearly show that the holistic approach provides better results. On average, basing on the considered use-cases the sum of the response times of the holistic approach is lower by 53.33%.

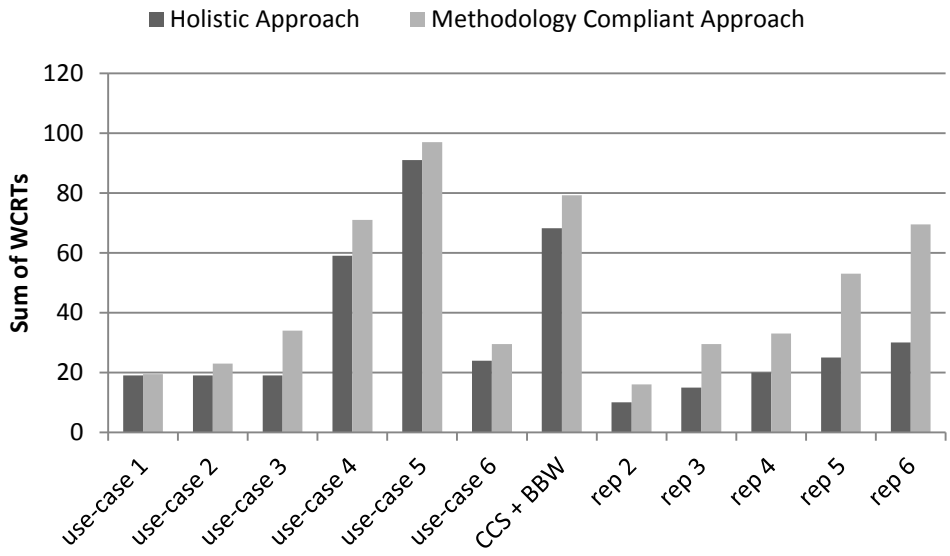

Figure 4.30. Comparison between the results of the MCDT and the Holistic Approach

The evaluation shows that lots of pessimism is being introduced when dividing the deployment on two stages, first with the allocation, second with the partitioning and scheduling. Impossibility to model the OS tasks and messages at the EAST-ADL2 level and in consequence inability to run the schedulability analysis test has a significant impact on the optimization approaches. Usage of the utilization metric as a substitute for the end-to-end responses metric, at the first stage, doesn't lead to the allocation for which an optimal solution for scheduling and partitioning can be found. Optimal in this case is understood as the optimal solution that can be found when considering the problem holistically. Of course staged approaches have an advantage of being more scalable as the subsequent problems are less complex. This capability was used when introducing the Two-Steps Approach (see 4.5). However in this case it is possible to run the schedulability test at each stage of this approach, hence the final metric was considered throughout the whole process. Authors of [63] in addition to the utilization metric defined another abstraction for the end-to-end responses metric, called preemptions metric. In principle it tries to minimize the possible amount of preemptions between the atomic functions. Nevertheless its

usage at the first stage, similarly when considering utilization, doesn't lead to the satisfactory results.

The provision of new deployment techniques, accounting for the functional entities (runnable entities) was the main objective so far of the chapter 4. They represent an intermediate step to accomplish the initial goal, i.e. the technique for the specification of time budgets. The last is called in this work TTBA (Technique for Time Budgets Assignment) and is described in the next section.

4.7. Time Budgets Assignment

The general idea of the TTBA is to interleave the deployment with the time budgets assignment. This is due to the bidirectional dependency between these two processes. Namely to lead a qualitative deployment as is the case for the techniques presented in this work, it is necessary to have an information about the WCETs of the runnable entities and for those whose WCET is unknown, specification of the time budgets. However if the architecture is already deployed it is easier to provide a better estimates of time budgets. For example knowledge about the allocation when assigning the time budgets for two communicating runnables, enables to determine whether an additional time budget needs to be reserved for the inter-ECU communication, if the two runnables are hosted by different ECUs. This means that the budget allocated for the communication will have to be subtracted from the time budget of these communicating runnables or from any runnable belonging to the same transaction/path in order to respect the end-to-end constraint. As will be shown in the related work the existing techniques either assume that the deployment is already fixed (which means it wasn't done considering the timing constraints) or the deployment is not known and hence the time budgets assignment doesn't take into account possible overhead for the communication. This section presents the first approach which tries to overcome this issue. Ultimately the problem tackled in this subsection consists of four sub-problems **(1) allocation, (2) partitioning, (3) scheduling** and **(4) time budgeting**, where the last concerns the relaxation of time budget values. Providing for a time budget that allows for additional slack time mitigates the design risks associated with uncertainties about the execution time of the runnable implementation delivered by the supplier. The authors of [64] propose a method to derive a certainty of obtaining a feasible system configuration under the assumption of uncertain design parameters such as WCET of new runnables. For this purpose, they define an uncertainty function that enables the system integrator

to estimate the risk of obtaining an infeasible design and consider it in the definition of the contract with a supplier.

These four sub-problems have significant cross-dependencies. Ideally, they should be solved as an integral problem, but this could be very challenging in terms of the computational effort that is required. Alternatively, they can be solved in stages, with the possibility of early choices restricting the set of available decisions for later stages. This work tries to lessen the problem by wrapping the staged solution in an iteration loop, in which the first stage is performed several times trying to improve on the results of the previous cycle. In fact the general idea is similar to the solution employed in the Two-Step Deployment Approach (see section 4.5) which uses two strategies; iterative improvement and divide and conquer.

In the following subsection some formalism will be added necessary to describe the TTBA. Next is the related work. Subsection 4.7.3 is a specification of the TTBA. Finally, the proposed approach is evaluated by presenting a set of results obtained by running the TTBA.

4.7.1. Formalism

This subsection extends the formalism introduced in the previous sections. The new concepts are gathered in the Table 4.8.

Concept	Definition
tb_{r_i}	Time budget for the runnable entity r_i. Time budget in our case represents the constraint imposed on the WCET of a runnable entity. In the work [65] it is called execution time budget. For the sake of simplicity the term "time budget" will be used. Please note that time budget, in opposite to the WCET, is not represented as a vector. This is due to the fact that time budget is specified for particular deployment hence at this stage the hosting ECU for runnable r_i is known.
$tb_{r_i}^m$	Minimal time budget for the runnable r_i. The designer has the option to provide a minimum value for tb_{r_i} as $tb_{r_i}^m$. Its intuitive meaning is a preliminary evaluation of the minimum required execution time for the functionality, based on the experience of the designer. If it is not specified, then $tb_{r_i}^m = 0$.
$tb_{r_i}^M$	Maximal time budget for the runnable r_i. Its intuitive meaning is a preliminary evaluation of the maximal execution time for the functionality, based on the experience of the designer. If not explicitly set, it is assigned with the period of the transaction to which r_i belongs.

RB	Set of runnables for which a budget assignment must be provided.
TBA	This set represents specific Time Budget values Assignment, i.e. the valuation. Namely each element is a value assigned for a corresponding time budget tb_{r_j}.

Table 4.8. Additional Formalism for Time Budgeting

4.7.2. Related Work

The problem of defining time budgets for components and runnables is affine to the issue of end-to-end deadline partitioning. Several research works have investigated the option of partitioning the end-to-end deadline into time windows or intermediate deadlines, upon the assumption that the interaction model allows the composition of the local response times to compute end-to-end response times. A graph-based algorithm for deadline partitioning to maximize the minimum slack is presented in [66], and an approach for periodic processes in [67]. More recently, deadline partitioning schemes for transaction chains scheduled under EDF or fixed priority can be found in [68], [69], [70]. Other efforts have been specifically tailored to automotive architectures. The TIMMO-2-USE project [29] discusses the need for time budgeting in the context of the process stages dedicated to the refinement of the system architecture. The project deliverables discuss a set of guidelines for budgeting the worst-case response times, based on the designer experience and do not provide a specific algorithm. Scheickl et al. [19] considers a similar process in which the definition of the WCRT budgets is based on the experience of the designer. Similarly, WCRTs are budgeted in [71], with a discussion on how different activation patterns (event- or time-driven) influence the specification of time budgets. [71] also studies the influence of time budgeting on the later reuse of ECUs, on the attempts to add new functions, or on the changes of the topology. Nevertheless, as in the previous two works, the approach relies on the experience of the designer to specify the time budget values.

The methodology of partitioning deadlines on response times is more suitable to the concept of federated automotive architectures [72], when suppliers provide hardware units or ECUs (Electronics Control Units) with operating systems and tasks, or at the very least when the responsibility of the task design is delegated to the suppliers. In the new concept of *integrated architecture*, enabled by AUTOSAR, the definition of the tasks and the design of the hardware architecture pertains to the integrator. Therefore, budgeting should be performed at the level of the WCET of the runnables.

The concept of time budgeting in the integration of automotive systems is among the research topics of the ALL TIMES project [65]. The approach proposed in the project deliverables takes as an input an already deployed architecture, i.e. the software components are already assigned to tasks and mapped onto the hardware platform. Since WCETs are not known, the deployment choices (assumed as predetermined and not subject to optimization) could very well be suboptimal and affect the final result (the assigned budgets). In [73] the budgeting problem is formulated and solved by applying Parametric Linear Temporal Logic (PLTL). A method is presented to automatically decompose end-to-end deadlines into a set of time budgets. The authors automatically compute a set of linear constraints for which they finally find a valuation (using a solver) that guarantees all deadlines and maximizes the values of the time budgets. The proposed solution also integrates the consideration of non-functional properties related to the ECU utilization [74]. As in all previous cases, the authors assume that the deployment, i.e. the integration of the software architecture with the hardware platform and the design of the task, is already done. In a fully integrated AUTOSAR solution, it is possible to leverage the freedom in the definition and allocation of the tasks to further improve the budget values.

The Table 4.9 gathers the related work and presents it in regards to the four criteria. These tell whether related approach requires designer knowledge (is manual) or not (is automatic), if it budgets WCRTs or WCETs, if it assumes that the deployment is unknown, and finally whether it directly relates to the automotive domain.

Work	**Manual/ Automatic**	**Budgeting WCRT**	**Budgeting WCET**	**Unknown Deployment**	**Automotive Context**
Di Natale [66]	Automatic	✓			
Gerber [67]	Automatic	✓			
Hong [69]	Automatic	✓			
Jayachandran [70]	Automatic	✓			
TIMMO2USE [29]	Manual	✓		✓	✓
Scheickl [19]	Manual	✓		✓	✓
Feiertag [71]	Manual				✓
Dixit [73]	Automatic	✓			✓
AllTimes [65]	Automatic		✓		✓

Proposed Approach	Automatic		✓	✓	✓

Table 4.9. Summary of the Related Work for Time Budgeting

4.7.3. Method for Time Budgeting

The Technique for Time Budgets Assignment (TTBA) builds upon three important assumptions:

1) time budgets represent the constraint imposed on the runnables WCET

2) the input for the TTBA these are the software and the hardware architecture which are not yet synthetized

3) the objective is to relax the time budgets maintaining at the same time the preservation of end-to-end deadlines.

Therefore the TTBA interleaves the time budgeting with the deployment process. By interleaving the two and because these are orthogonal concerns the relaxation of the time budgets can be controlled by the end-to-end deadlines. Please also note that the response times can be computed only for a deployed architecture hence the time budgeting needs to be done in parallel with the deployment process. Lack of a deployed architecture is the reason why some of the current approaches are budgeting worst case response times.

This work elaborates on two heuristic approaches for finding time budgets. The first (One-step TTBA) provides a holistic one-step solution to the problem, whereas the second (Staged TTBA) divides it into two sub-problems solved one after the other. In any case both of these approaches reuse the previously presented deployment techniques with few minor changes (both in case of the One-step and Staged TTBA) to consider the budgeting. The following three paragraphs present the optimization objective that induces the relaxation of time budgets, the One-step TTBA and finally the Staged TTBA.

Optimization Objective

This work considers an optimization metric expressing the relaxation of time budgets within the end-to-end deadline constraints. The function $f_{tb}(TBA)$ in eq. 4.47 requires as an input the set TBA of runnables with the specific valuation for their time budgets. It is defined as the minimum time budget value for all runnables in RB normalized with respect to the target range $(tb_{r_k}^m, tb_{r_k}^M)$.

The optimization objective is to maximize $f_{tb}(TBA)$, or equivalently, to maximize the minimum normalized time budget among runnables in RB.

$$\mathrm{f}_{tb}(TBA) = \min_{r_k \in RB} \frac{tb_{r_k} - tb_{r_k}^m}{tb_{r_k}^M - tb_{r_k}^m} \quad \textbf{4.47}$$

Relaxation of budget values lies in the interest of the [65]. The [65] employs the binary sensitivity analysis [75] which searches for an upper bound of the runnable execution time so that the system remains schedulable. The proposed algorithm is designed to consider the relaxation of only one runnable, which is why the metric of interest does not need to be normalized. In our case, relaxation should affect all the runnables in RB. [76] accounts for more than one runnable,

by simply returning a schedulability region of all the possible combinations of the time budgets for runnables in RB. Our proposed metric in 4.47 instead targets at only the best combination, i.e. one that equally distributes the budget constraints among different suppliers delivering implementation of runnables from RB.

One-step TTBA

To solve the problem of time budgeting (as defined consisting of four sub-problems) this work first considers a one-step approach and a solution based on a Genetic Algorithm and MILP. A genetic algorithm is used to find solutions for the deployment problem which includes allocation, partitioning, and scheduling (the first three sub-problems). Simply here this work is reusing the GA as defined before. In particular it refers to the GA solving the deployment holistically as described in section 4.4.3 for data driven activation model following some minor adjustments. Also, based on this version, the tests presented in the section 4.7.4 were run. However as the changes are not significant the TTBA can easily port the GA used for the deployment of time-driven systems (see subsection 4.4.5) or the GA implementing the Two-Step Approach (section 4.5). The last in fact would be desirable in order to further improve on the scalability of the TTBA. This issue is discussed in the evaluation part (subsection 4.7.4).

The principle of the One-step TTBA is to assign time budgets for all the possible deployment configurations created during the run of the GA. The deployment configuration in the context of the GA implementation is represented with a single chromosome. The time budgets for each deployment (chromosome) are assigned using the Time Budgeting Algorithm (Algorithm 2)

explained later in some more details. This means that the first change to the specification of the GA as defined in subsection 4.4.5 is that Algorithm 2 is run for each chromosome. The second difference lies in the fitness function which changes as the optimization objective this is now the relaxation of time budgets. Hence chromosomes are ranked according to the result of the metric function from the eq. 4.47. This function requires as an input TBA, i.e. specific time budgets valuation, which is obtained by running the Algorithm 2. Of course at the initial stage of the GA most of the chromosomes will represent deployments for which any relaxation of time budgets will not be possible. This means that for all of them their fitness value would equal 0. This is not the best solution as it won't lead to the improvement of the initial population. Therefore to differentiate between the chromosomes for which fitness equals 0 (i.e. $ch_i(f_{tb}(TBA)) = 0$), they are evaluated in respect to how much they violate the end-to-end deadlines. It simply means that to their fitness (equal 0) additional factor will be added (as presented in eq. 4.48) which will always represent a negative value.

$$\bigwedge_{ch_i(f_{tb}(TBA))=0} ch_i\big(f_{tb}(TBA)\big) = \sum_{R_{\Gamma_j} > D_{\Gamma_j}} \frac{D_{\Gamma_j} - R_{\Gamma_j}}{D_{\Gamma_j}} \qquad \textbf{4.48}$$

Time Budgeting Algorithm

Within the GA optimization cycle, Algorithm 2 is executed for each chromosome to compute the corresponding optimum set of time budgets based on which value for the metric function $f_{tb}(TBA)$ can be calculated.

The time budgeting algorithm has four inputs: Ψ, RB, ϵ and MRB. ϵ is the maximum error on the computed budgets that controls the terminating condition (line 15). The lower is the value of ϵ, the more accurate are the time budgets, and the larger is the runtime of the algorithm. M_{RB} is a set of upper bounds on the runnable budgets computed for a specific deployment Ψ, where $M_{RB}(i)$ is the maximum value for r_i. The values in M_{RB} are computed before running Algorithm 2, based on the end-to-end deadlines, utilization bounds, and the constraints $tb_{r_i}^m$ and $tb_{r_i}^M$. The formulation that is used to compute the bounds M_{RB} is discussed after the description of the time budgeting algorithm (section 3.6).

Algorithm 2 tries to relax the time budgets for all the runnables in RB according to the metric 4.47 using a binary search algorithm (as in the sensitivity analysis test in [75]). The upper bound values are tried first, giving the maximum possible value of $f_{tb}(TBA)$ (lines 6-8). If the

corresponding configuration is schedulable, it is returned as the optimum value (line 9). If not, then the algorithm assigns to each r_j in RB a budget value that is the medium value between the minimum $tb_{r_j}^m$ and the upper bound $M_{RB}(i)$ (lines 11-13).

From this point on, Algorithm 2 continues by iteratively reducing the range of the time budgets, defined as $[Ltb_{r_j}, Utb_{r_j}]$ for runnable r_j. The algorithm works as a binary search. In each iteration, if the current budget values, at the midpoint between the upper and lower bounds result in a schedulable solution, the upper bound Utb_{r_j} remains the same, and the lower bound Ltb_{r_j} is updated to be midpoint (line 20), and the range is reduced to be half of the size. If the current settings result in an unschedulable solution, it means that the time budget value is too large, and the next iteration will search within the lower half of the range (line 22).

In [75], budget values are computed for each runnable separately, in a set of recurrent calls, exploring all the possible options for the relaxation of each individual runnable budget, at the price of higher complexity. However, for the metric 4.47 this is not required. Given any optimal solution according to 4.47, there exists another solution with the same value of 4.47 that is computed by our bisection algorithm, performing an equal relaxation of all time budgets (i.e. proportionally to $tb_{r_i}^m$ and $tb_{r_i}^M$). Of course, the solution computed by Algorithm 2 can have smaller budget values for those runnables that are not affecting the value of 4.47.

Algorithm 2: Time Budgeting Algorithm

Require: Ψ, RB, ϵ, M_{RB}

1: **forall** $\boldsymbol{r_j} \in RB$ **do**
2: $Utb_{r_j} = \mathrm{M}_{RB}(j)$
3: $Ltb_{r_j} = tb_{r_j}^m$
4: **end forall**
5: **if** *isSchedulable*(Ψ, M_{RB}) **then**
6: **forall** $\boldsymbol{r_j} \in RB$ **do**
7: $TBA.set(r_j, M_{RB}(j))$
8: **end forall**
9: **return** TBA
10: **else**
11: **forall** $r_j \in RB$ **do**
12: $tb_{r_j} = (Utb_{r_j} - Ltb_{r_j})/2$
13: **end forall**
14: **end if**
15: **while** $\exists_{r_j \in RB} Utb_{r_j} - Ltb_{r_j} \geq \epsilon$ **do**
16: **forall** $\boldsymbol{r_j} \in RB$ **do**
17: $TBA.set(r_j, tb_{r_j})$
18: **end forall**
19: **if** *isSchedulable*(Ψ, TBA) **then**
20: **forall** $r_j \in RB$ **do**
21: $Ltb_{r_j} = tb_{r_j}$
22: **end forall**
23: **else**
24: **forall** $r_j \in \mathrm{TB}_r$ **do**
25: $Utb_{r_j} = tb_{r_j}$
26: **end forall**
27: **end if**
28: $tb_{r_j} = Utb_{r_j} - (Utb_{r_j} - Ltb_{r_j})/2$
29: **end while**
30: **return** TBA

Algorithm 2. Algorithm for the One-dimensional Binary Search

Calculating M_{RB} using MILP

Finally, the upper bounds M_{RB} that are required to reduce the initial interval of possible budget values in Algorithm 2, are computed using MILP formulation. The values in M_{RB} are (optimistic) upper bounds and do not guarantee the system schedulability as the constraints used for their computation are a linear approximation representing only a necessary schedulability condition that does not consider interference. However, they are useful in constraining the search space for the bisection algorithm.

In the MILP formulation, the problem is represented with parameters, decision variables, and constraints over the parameters and decision variables. Moreover, an objective function is defined to characterize the optimal solution.

Variables: the only set of variables is $M_{RB}(r_i)$ where $r_i \in RB$.

Objective function: the objective is to maximize the metric function in Equation 4.47, with $M_{RB}(r_i)$ in place of tb_{r_i}.

Constraints: three types of constraints are considered:

- Utilization constraints – utilization bound applies to each ECU (u_{e_i}). If not otherwise specified, the limit value is 1.

$$\forall_{e_i \in E} \sum_{e(r_j)=e_i \wedge r_j \in RB} \frac{M_{RB}(r_j)}{P_{\Gamma_{r_j}}} \leq u_{e_i} - \sum_{e(r_j)=e_i \wedge r_j \in R \setminus RB} \frac{C_{r_j,e_i}}{P_{\Gamma_{r_j}}} \quad \textbf{4.49}$$

- Computation time constraints - are a linear (under-) approximation of the deadline constraints, ensuring that the sum of the execution times and budgets on each chain is lower than the deadline. These constraints do not consider interference and therefore do not guarantee end-to-end deadlines.

$$\forall_{\Gamma_i \in \xi} \sum_{\Gamma(r_j)=\Gamma_i \wedge r_j \in RB} M_{RB}(r_j) \leq D_{\Gamma_i} - \sum_{\Gamma(r_j)=\Gamma_i \wedge r_j \in R \setminus RB} C_{r_j,e(r_j)} \quad \textbf{4.50}$$

- Minimum and maximum value constraints

$$\forall_{r_i \in RB}\ tb_{r_i}^m \leq M_{RB}(r_i) \leq tb_{r_i}^M \quad \textbf{4.51}$$

Staged Approach

The one-step holistic approach is simple and effective but does not scale to very large-size problems. Hence, an alternate solution was developed by dividing the four sub-problems in two stages. The first stage solves the first three sub-problems on deployment (including placement,

partitioning and scheduling). The second stage tries to optimize the time budgeting only. The two stages are computed sequentially inside a loop until there is no further improvement as shown in Figure 4.31. The computation time savings derive from the execution of Algorithm 2 once for each iteration instead of once for each chromosome.

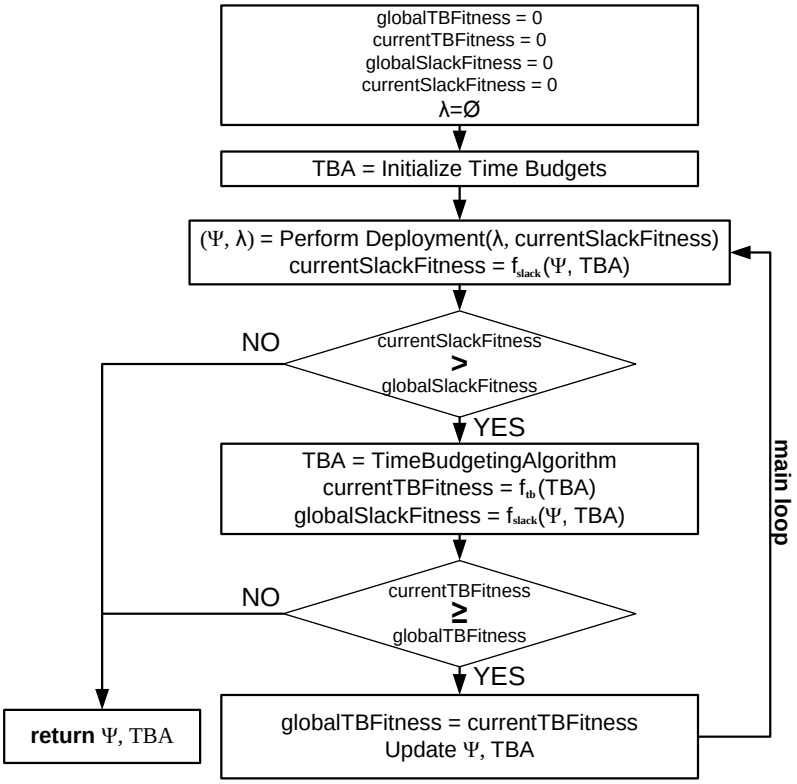

Figure 4.31. Iterative Improvement Loop for the Staged Approach

The staged algorithm implements an iterative improvement strategy that is in essence a local search. Starting from an initial solution, the current best solution is tentatively improved in the iterations of an inner cycle that includes the two optimization stages. If at any iteration, the two stages fail to produce a better result, the algorithm terminates and returns the best solution found until that point. The algorithm starts with the initialization of the variables storing the population of chromosomes λ, and the current best metric values (first two blocks from Figure 4.31). The second stage (second block in the figure) initializes the values of time budgets with their minimum values, i.e. $\forall_{r_i \in RB} tb_{r_i} = tb_{r_i}^m$. The rationale for this choice is that we do not want the algorithm (a local search) to end prematurely and we try to ease schedulability (and provide for

maximum allocation freedom) as much as possible in the first step. In the experiments section, we discuss the impact of different values for the initial time budgets.

Next, the deployment optimization is performed considering the current values of the time budgets (the first time the loop is entered these are the initial budgets). During the deployment optimization stage, budgets are fixed, the metric function $f_{tb}(TBA)$ has a constant value, and cannot be used to evaluate the quality and drive the selection of the deployment solutions. Hence, the fitness function considered in this step is based on the end-to-end response times. The function $f_{slack}(\Psi, TBA)$, defined in equation 4.52, expresses the goal of maximizing the minimum slack time (the difference between the deadline and response time) of each transaction.

$$f_{slack}(\Psi, TBA) = \min_{\Gamma_i \in \xi}[D_{\Gamma_i} - R_{\Gamma_i}] \qquad \textbf{4.52}$$

Maximizing the minimum slack means maximizing the minimum distance between the response time and deadline of any transaction, which is an indication of an opportunity for having larger budgets and hence a better deployment. The function $f_{slack}(\Psi, TBA)$ is computed based on the schedulability analysis formulas for computing the response times of runnables and messages. Subsection 4.7.4, discusses the results obtained when trying different metric function at this stage.

At each iteration of the main loop, a new deployment solution is computed and then evaluated. If the value of $f_{slack}(\Psi, TBA)$ does not improve on the current best solution, i.e., the minimum slack is lower, the loop terminates and the best solution computed up to this point is returned. Otherwise, Algorithm 2 is executed to compute a new optimum set of time budgets (for the current deployment). Then, the fitness value $f_{tb}(TBA)$ is computed for the new set of time budgets and the new fitness value is compared with the current best. If the new deployment/budgets improve on the current best solution, they are considered for the next iteration, and the new budget drive the next deployment optimization step. Otherwise, the algorithm terminates and returns the best current deployment and time budget solution. The procedure is summarized in Figure 4.31.

Besides the different optimization metric used during the deployment, another significant difference with the One-step TTBA deployment algorithm is the stop condition. The loop terminates not only if no improvement is found after *n* internal GA iterations, but also when the

fitness of the best chromosome from the new population is not better (lower or equal) than the current best fitness value (*currentSlackFitness* ≤ *globalSlackFitness*).

Finally, the set λ defines the initial population for the GA algorithm. At each iteration round, λ preserves the population selected in the previous run of the deployment algorithm. When the deployment is run for the first time and the set λ is empty, the deployment procedure initializes λ with a random initial set of chromosomes which is consistent with the constraints that apply to the system configuration.

4.7.4. Evaluation & Conclusions

To evaluate the algorithms, a series of experiments have been performed on a collection of case studies. First, a set of case studies is used to compare the one-step approach with the staged approach in terms of the quality of the results and the required execution time. Next, this subsection discusses the robustness of the staged approach by evaluating the influence on the final results when replacing the metric in eq. 4.52 with a different function. Finally, we present experiments to see if and by how much different initial assignments of time budget values affect the final result. All tests were run on a machine with 8GB of memory and a single processor running at 2.4GHz. Also, all the tests assume the maximum error factor ϵ=0.5 and the stop condition for the GA (regardless of initial population size) is that n=30 consecutive iterations compute the same value.

Representative Use-Case

The evaluation part starts first with a representative use-case to show the principles of the TTBA. It is a use-case combining the CCS (Cruise Control System) and ABS (Anti-lock Braking System) from the Figure 4.8, used previously for testing the deployment techniques. The functional model contains twelve runnables in four transactions with their deadlines and trigger periods. For five runnables, i.e., *Input Acquisition*, *Input Interpretation*, *Basic Function*, *Diagnosis* and *Self Diagnosis*, the WCET information is not available and time budget must be assigned (they belong to the set *RB*). The other seven runnables are assumed as reused from legacy libraries and their WCETs are known. The hardware topology contains four ECUs, each connected to the single CAN bus. The Table 4.10 displays minimum and maximum budget values for the runnables in *RB*. It also shows their computed time budget values and for runnables

which don't belong to *RB*, their WCET information is provided. The Figure 4.32 displays the deployment configuration for the CCS+ABS example obtained with the One-step TTBA.

Runnable	WCET	tb^m	tb^M	tb	τ	ECU
Input Acquisition	-	0	40	8.73	τ_1	1
Input Interpretation	-	0	40	8.73	τ_1	1
Basic Function	-	0	40	8.73	τ_1	1
Diagnosis	-	0	10	2.18	τ_1	4
Self Diagnosis	-	0	10	2.18	τ_2	1
Limp Home	1.03	-	-	-	τ_1	4
Speed Setpoint	3.5	-	-	-	τ_1	1
Application Condition	3.92	-	-	-	τ_1	1
Controller	1.4	-	-	-	τ_1	2
Data Processing	10	-	-	-	τ_1	3
AL1	15	-	-	-	τ_2	2
AL2	15	-	-	-	τ_2	4

Table 4.10. Results for Time Budgets Assignments and Initial Constraints

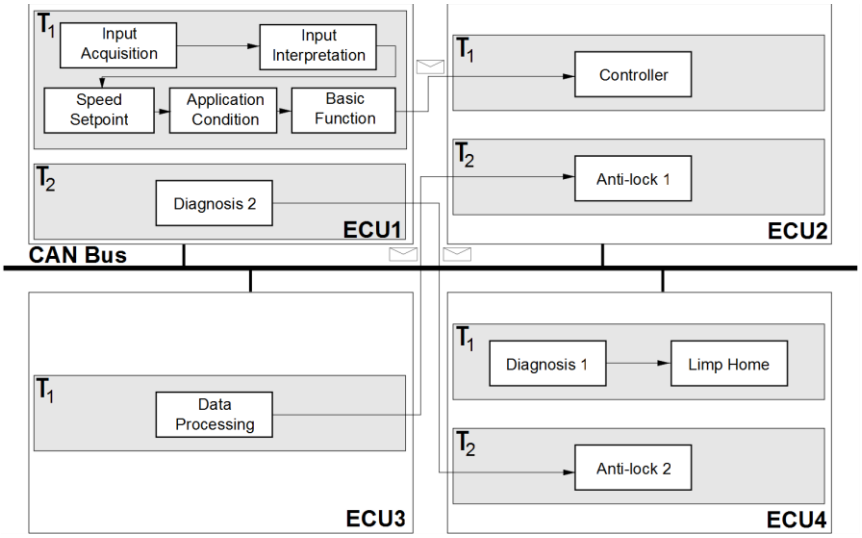

Figure 4.32. Deployment Configuration for CCS and ABS

One-step vs. Staged TTBA

This part presents results of the comparison between the one-step holistic algorithm and the staged iterative approach. It examines and compares the quality of the solutions obtained with the

two approaches, i.e. the final fitness values - the maximized $f_{tb}(TBA)$ and the runtimes that are required by the two algorithms.

For this purpose, the automotive case study has been extended with lower and higher complexity examples that have been generated starting from the 50-runnable case study (index 9 in a list) presented in [47] and extended to lower and higher sizes. Table 4.11 shows a summary of the fifteen system configurations by growing complexity. The table contains in the first column an index identifying the test case, in the second column the total number of runnables, in the third column the number of runnables for which budgets must be assigned, and, finally, the number of ECUs in the hardware architecture. In all these examples, we assume a single CAN bus connecting all ECUs. The original automotive case study is in the fourth row, with twelve runnables (as shown in the second column) and 7 of them in *RB*. Finally, as a further assumption, each software component has only one runnable entity. This means that for each runnable, its placement is independent of any other runnable's placement, as AUTOSAR requires that all runnables in the same component must be placed to the same ECU.

Test nb	**Runnables**	$\|RB\|$	**ECUs**
1	5	2	2
2	6	2	2
3	10	4	4
4	12	5	4
5	16	6	6
6	20	8	8
7	32	12	9
8	40	14	9
9	50	17	9
10	60	20	9
11	70	25	10
12	80	27	12
13	90	30	16
14	100	35	18
15	200	70	36

Table 4.11. Properties of the Testing Input Architectures

Figure 4.33 shows the final fitness value of the best solution obtained by the One-step TBBA, compared with the Staged TTBA. The size of the initial population of the GA considered for these tests is 10000. The fitness value of the solutions computed by the two approaches for tests 1 to 6 is exactly the same, and also the same value was computed for test 10. For tests 7 to 9 the staged approach provided results slightly better than the one-step algorithm (in detail, 0.33%, 0.34% and 3.48% better, respectively). Finally, the one-step approach could not compute the final solution for case 11 after more than 24 hours of processing time. During this time, the GA internal loop performed 76 iterations. The best result obtained after this time was 8.75% worse than the final result computed by the staged approach (after 5.86 hours). The Staged TTBA was also tested on a case (test 12) with 200 runnables, 70 runnables in *RB* and 36 ECUs. The best result (fitness value of 0.13) was reached after 28.7 hours. The processing time required by the one-step algorithm prevented a realistic comparison in this case.

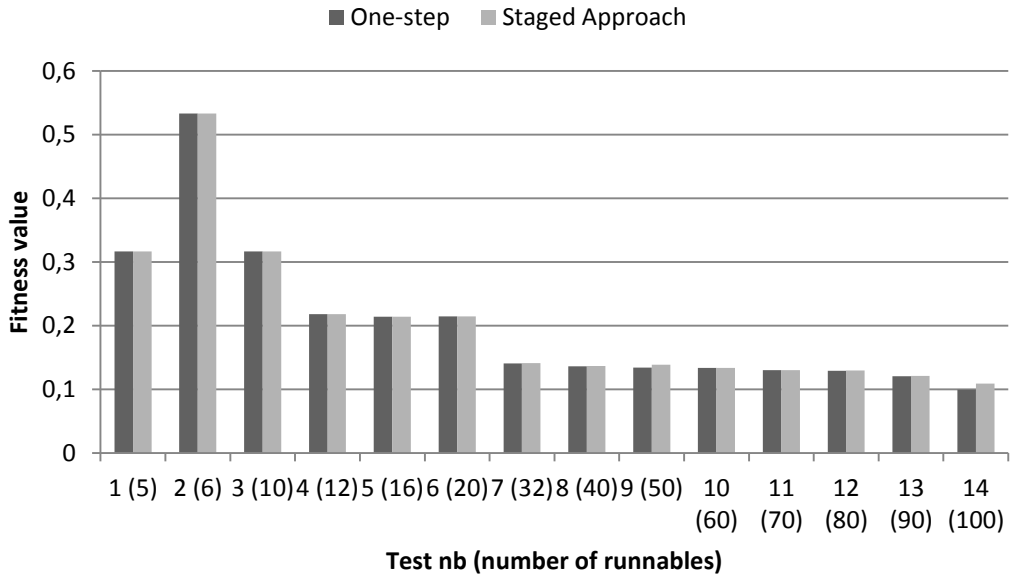

Figure 4.33. Results for One-step and Staged Approach (GA Initial Population = 10000)

Not only the staged approach manages to get equal or better quality solutions than the one-step approach, but computes them in a much shorter time. The graph in Figure 4.34 shows a comparison of the execution times required by the two algorithms for each experimental case. The runtime of the one-step and staged approaches increases not only with the problem size but also with the size of the GA initial population. Augmenting the size of the GA population is desirable, as in many cases this leads to a better value for the final solution. In our experiments,

the final fitness value was mostly independent from the size of the initial population if it has more than 1,000 initial chromosomes. The runtimes in the figures are shown for an initial population of 10,000 chromosomes. As shown by the graphs, the two algorithms have an execution time that still grows exponentially with the size of the problem. However, the staged approach can solve problem configurations of a size comparable with the typical problems of the industry (approximately 6 hours for 100 runnables).

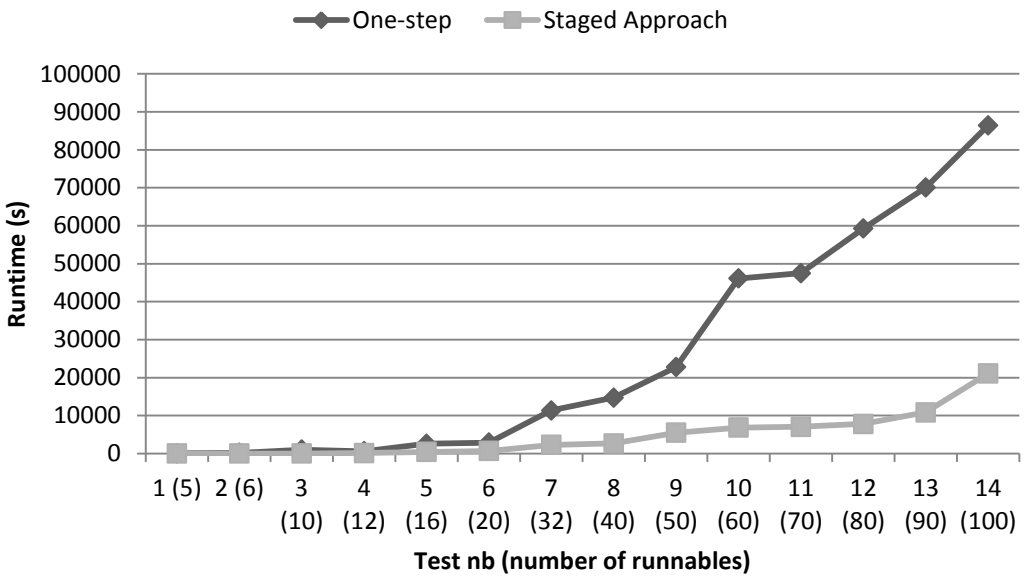

Figure 4.34. Runtimes of One-step and Staged TTBA (GA initial population = 10000)

Robustness of the Staged TTBA

This subsection evaluates the sensitivity of the staged approach with respect to the metric used in eq. 4.53 to select placement solutions. As an alternative to maximizing the minimum laxity metric in (eq. 4.52) this work used a metric function (eq. 4.53) that minimizes the sum of the latencies of (a subset of) the transactions (which is another indication of an opportunity for assigning larger budgets).

$$f_{e2e}(\Psi, TBA) = |\xi| - \sum_{\Gamma_l} \frac{R_{\Gamma_l}}{D_{\Gamma_l}} \qquad \textbf{4.53}$$

As shown in Figure 4.35, the original metric (4.52) provides better optimization results on the tests from 1 to 11, in the average by 4.05%. The reason for this is intuitive. Metric 4.53 may lead

to situations, in which for some transactions the response time is significantly reduced, whereas for others it is close to the deadline. This last set of transactions is a bottleneck for the relaxation of time budget values that follows next. This is not the case for the metric 4.52 which minimizes the response times with respect to the deadlines and maximizes the minimum value (does not operate on a sum of values).

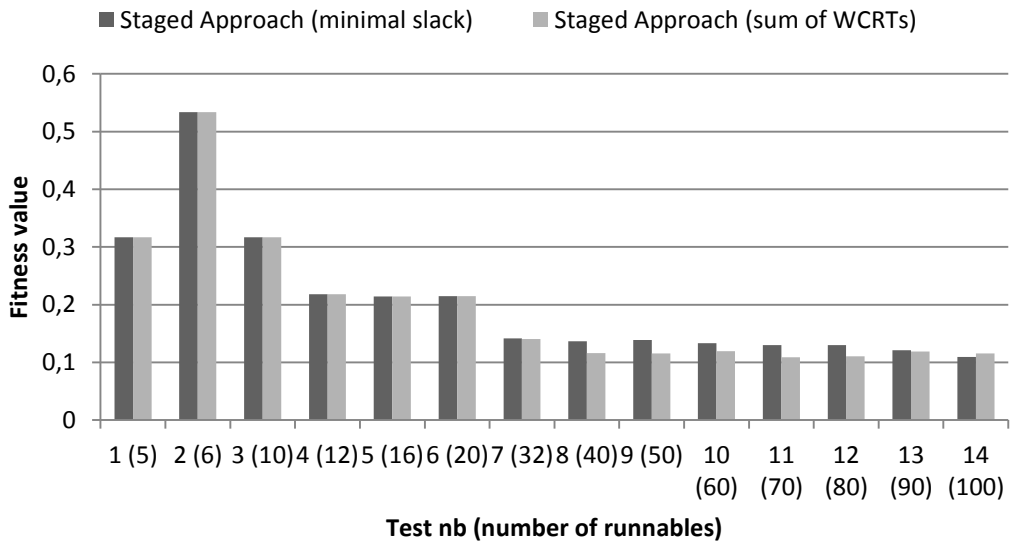

Figure 4.35. Comparison of two different metrics for Staged TTBA

Concerning the runtime, there is no significant difference between the two metrics. The slight differences in runtimes are mostly caused by the difference in the number of iterations of the main loop in the staged approach. However, the number of iterations (see Figure 4.36) is mostly similar and so are the runtimes. The only exception is test 11 where the use of metric 4.53 resulted in 138 iterations and a runtime of 7.6 hours, which is 29.75% higher than the case of a slack metric (4.52), but the final result is slightly better.

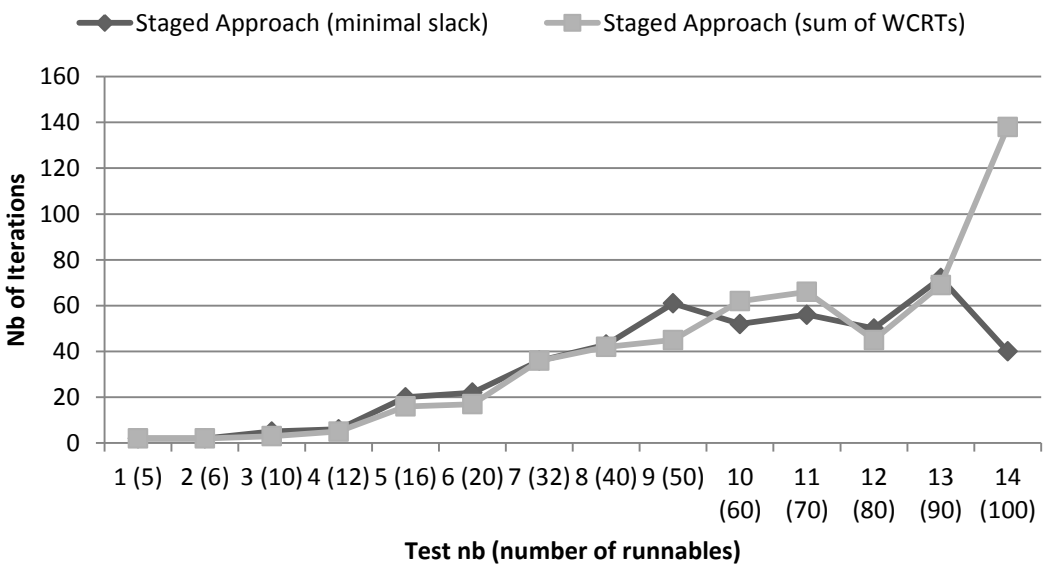

Figure 4.36. Comparison of Number of Iterations of two different metrics for Staged TTBA

Influence of Initial Time Budgets

Additional tests were run to confirm the choice of starting the iterative improvement algorithm with an initial setting of time budgets equal to the minimum allowed value for each runnable in *RB*. As previously stated, since the algorithm is a local search and terminates when no further improvements are possible, a selection of initial values that prevents schedulability would likely cause a premature termination. To verify, initial budget assignments at 1%, 2%, 3%, 5%, 10% and 20% of the range between the minimum and maximum values $[tb_{r_i}^m, tb_{r_i}^M]$ were tried. In all experiments, there was no sensible difference in the quality of the final result. However, for the highest values of the initial budgets (a 20% increase over the minimum value), the algorithm ended prematurely for all cases from 7 to 10. In these cases, the first deployment step from the iterative algorithm was not able to find any feasible solution.

Comparison with All-Times Approach [65]

This part compares the presented approach with the work on time budgeting coming from the All-Times project [65].

Budgeting Algorithms

First was studied the performance of the presented time budgeting algorithm (see Algorithm 2) with respect to the algorithm used in [65]. Authors of [65] claim that during each manual

reconfiguration of deployment, they use the sensitivity analysis to find the relaxation of time budget values. The sensitivity analysis they refer to comes from [75]. This analysis is applicable if there is only one runnable in RB. The extended version of this algorithm that can budget multiple runnables is deifned in [76]. Ultimately this is the algorithm implemented in this work for the comparison. It was then ported in the one-step and staged approach for deployment instead of the Algorithm 2 to see if it can be applied in the automated process of budgets and deployment specification.

Table 4.12 presents the runtimes which clearly shows that usage of Algorithm 2 leads to shorter runtimes in the context of both one-step and staged approaches. This difference is more significant for the one-step approach because the calls to the budgeting algorithm occur much more often. In fact, starting from test nb 3, usage of algorithm from [76] in the context of the one-step approach was too time consuming, i.e. after 24 hours the algorithm did not finish executing. The reason is that sensitivity analysis from [76] was designed to construct the map of all budget combinations for which the system remains schedulable (the schedulability region). Our algorithm is adapted to the metric of interest which allows to select one single configuration of time budgets, which equally distributes budget constraints among the suppliers based on the predefined values of $tb_{r_i}^m$ and $tb_{r_i}^M$. For all the tests for which algorithms terminated, we obtained the same fitness value, which supports the usage of the Algorithm 2 in the automated process of time budgeting and deployment.

Test nb	One-step	Staged	One-step with [76]	Staged with [76]
1	68.474	3.874	2088.665	24.416
2	177.383	4.667	2006.573	31.96
3	1014.547	27.442	>86400	699.729
4	598.003	131.609	>86400	1072.178
5	2565.429	429.153	>86400	84132.114
6	2862.052	685.726	>86400	>86400

Table 4.12 Runtimes (seconds) of one-step and staged approach (GA initial population = 10000) when using different budgeting algorithms

Improvements due to deployment

Interleaving of deployment with time budgeting has positive impact on the relaxation of budget values. The approach from [65] assumes that the deployment is known a priori. On the other hand, our techniques (either one-step or staged) interleave the deployment with budgets specification. This additional design freedom allows to further improve on time budget values. To show the gain, we first fix the deployment for tests 1 to 6 using presented deployment technique (the runnables from RB were assigned WCETs equal to $tb_{r_i}^m$). Then the budgeting algorithm from [65] was run and the following values for metric in Equation (4.47) were obtained: 0.31666666, 0.53333336, 0.178125, 0.088630214, 0.0712207, 0.07819336. By further manipulation of deployment interleaved with budgets assignment, we managed to get results: 0%, 0%, 77.78%, 146.15%, 200.7%, 174.52% better for tests 1 to 6 respectively.

4.8. Conclusions

This chapter presented a set of techniques for the synthesis of the automotive architectures. Their main characteristic is the consideration of the functional entities as the base for the deployment. It was shown that most of the current approaches don't refer to the functional but implementation model as an input for the synthesis. This is contradictory to the current trends in the MDE for automotive which is to abstract from the implementation details and configure the system architectures earlier and hence account for the functional specification. As the scalability poses a big challenge in a delivery of the deployment techniques a heuristic based on the divide and conquer and iterative improvement strategies has been proposed. This chapter also exhibited a new idea for the specification of time budgets. Presented technique by combining deployment with time budgeting, grants the possibility for qualitative deployment even though the WCET information for certain runnables is absent. Finally it advocated refinement of the EAST-ADL2/AUTOSAR methodology to enable the holistic consideration of the deployment problem.

5. UML based & Optimization-aware modeling of the Automotive Architectures

The focal point of the previous chapter was a designation of the crucial design steps and the delivery of the techniques to take off the burden of the manual system configurations from a designer. Its content contributes to the principal goal of this thesis which is to add to the current efforts trying to employ the Model Driven Engineering (MDE) in the context of the automotive SW/HW architectures design. The presented techniques work on an abstraction level named the "system level" which represents a higher abstraction of a system specification. They don't require low level characteristics of a system such as the exact properties of a Basic SoftWare used on each of the ECUs. Consequently they are applicable on the system level models and as such can serve to configure them. Parallel development of techniques supporting the model analysis and optimizations is crucial for the employment of the MDE in any kind of industry, especially in the automotive domain. Also the other way around, modeling languages should embed concepts enabling to detail attributes needed to run the analysis/optimizations. Last issue but not least is the integration of the modeling analysis and optimization activities. The base requirement is the presence of the analysis/optimization enabling artifacts in the modeling languages. Secondly these are the transformations used to define the context for the analysis/optimization and how their results can be applied on the system models.

This chapter is focused on the integration problem. It starts by presenting the related work to show the deficiencies in the automotive domain concerning the frameworks that would integrate the modeling, analysis and optimization (MAO) activities. Next is the description of a framework proposed in this thesis, called AFfMAO (Automotive Framework for Modeling Analysis and Optimization) (Section 5.2). It is built as an instance of the Automotive Architecture Framework. Consequently sections 5.3 and 5.4 present the main viewpoints of the AAF related to the architecture specification analysis and optimization. They also discuss the final implementation of them within the AFfMAO through the UML profile mechanism. Section 5.5 is about the correspondence rules which enforce relations within an architecture description. Finally this chapter is concluded in the last section 5.6.

5.1. Related Work

The main objective of the related work is to show the deficiencies in the nowadays tooling/frameworks and practices for the model based development of the automotive architectures. In particular these are the:

1) Poor MAO integration. The rest of the mentioned deficiencies highly contribute to this problem.
2) Lack of the modeling capabilities to express the optimization concerns.
3) Absence of a standardized Architecture Framework for the automotive.
4) Weak interoperability between the tools.

Correspondingly this section reports first on the commercial and academia tooling used for the specification of automotive architectures, paying particular attention to their modeling, analysis and optimization capabilities if supported. Secondly as the advertised framework is based on the AF, related work discusses the concept of architecture framework for automotive. Finally it concludes by gathering the properties of interest and described tools in the Table 5.1 to highlight the main differences with the AFfMAO.

5.1.1. Commercial Tooling

Lots of tools based on the MDE principles have emerged to support software development of automotive systems. Many of them are based on the MDE principles using AUTOSAR or Simulink (Simulation and Model-Based Design) environment [77]. They mainly support the specification of system and network architectures and code generation. Manufacturers provide also software for testing embedded systems. For this purpose they use techniques such as HIL (Hardware in the Loop) [78] or SIL (Software in the Loop) [79]. The primary providers of tools supporting AUTOSAR for automotive system design are dSPACE [30], Vector [31] and Mentor Graphics [80]. All of them support AUTOSAR in their tools chain.

DSpace provides a product called SystemDesk [81]. Its main functionality is the specification of AUTOSAR software components, ports, and runnable entities. SystemDesk has interfaces to communicate with other tools such as TargetLink [82]. It is used for code generation based on specifications of software components. SystemDesk adopts solutions to aid system design in a distributed development environment. It allows subsetting of the system architecture to be used

by different subcontractors and also supports the reverse process, i.e., merging of multiple architectural subparts into a single AUTOSAR model.

Vector supports designers delivering tool called DaVinci Developer [83]. Their product, similarly to SystemDesk, provides clear, graphical representation for AUTOSAR software components and functionality to specify and integrate them. DaVinci Developer enables design of systems with single or many ECUs. For testing purposes, Vector offers DaVinci Component Tester. It can validate software components without hardware architecture, using AUTOSAR concept of VFB (Virtual Function Bus).

Mentor Graphics supports design of automotive systems via its VSx (Vehicle Systems) toolset [84]. This collection contains a variety of tools. The first tool in this toolset is VSA (Vehicle System Architect) for defining the overall system architecture. These are specifications of both, the software and hardware architectures, the ECU resources, and the system topology. From this it is possible to generate code automatically using BridgePoint [85]. This is code representing software components and their communications at the VFB (Virtual Function Bus) level. The VFB is an abstract communication environment. Next in the development chain is VSB (Vehicle System Builder). It consists of various plugins. They allow configuring the BSW (Basic SoftWare) and operating system through partitioning runnable entities in tasks. The VSI (Virtual System Integrator), similarly to the DaVinci Component Tester, analyzes software components at the VFB level. It is an execution environment for AUTOSAR systems providing early validation of software functionality on a virtual ECU and BSW. It is used to prove application software correctness even before hardware is available. Hardware-based testing is performed using VST (Vehicle System Tester).

The joint initiative team established by the members and affiliated partners of the AUTOSAR group defined another development platform for AUTOSAR compliant systems. Artop (AUTOSAR Tool Platform) [86] is an implementation of common base functionality for AUTOSAR development tools, based on Eclipse. The basic version of Artop allows only specification of the software and hardware architectures. The editor depicting the system is based on a hierarchical model browser and, therefore, does not provide a clear view of the overall architecture.

A current, important drawback of these tools (except for Artop) is the lack of support for AUTOSAR 4.0, which prevents timing requirements specification that is compatible

with the standard. This leads to the inability to carry out schedulability analysis. **For these reasons, none of the above tools are able to recommend optimal configurations for mapping software components onto ECUs or for mapping runnable entities onto OS tasks.** On these matters they rely solely on the experience of the system architect.

Another common drawback of these commercial tools is that the system descriptions rely too heavily on implementation-specific terms and concepts. They don't cover the feature, functional and design layers. Recent advancements of the EAST-ADL2 convinced certain tool suppliers to support this language. Currently the tools which make EAST-ADL2 modeling possible these are the MetaEdit+ from MetaCase [87] and SystemWeaver from Systemite [88]. Also the previously mentioned VSA from Mentor Graphics has been extended with the EAST-ADL2 modeling. Others like PREEVision [89] from Vector have their own modeling concepts related to functional level. This confirms the interest in moving with a design starting point to the abstraction layer which is higher than the software layer.

5.1.2. Academia Tooling

Apart from the commercial tools worth noting are the academic initiatives presenting prototypes supporting automotive system design. One of them is the AutoMoDe (Automotive Model-based Development) methodology [90] based on custom, problem-specific design notations with an explicit formal foundation. This approach has been prototyped within the existing framework called AutoFocus (A Distributed Multi-User CASE Tool) framework [91]. A step forward in their approach is introduction of higher abstraction levels adjusted to automotive design chain. Also in regards to the commercial tooling, AutoFocus offers computer aided design related to the deployment and its optimization in regards to the timing properties [92]. This is a very recent functionality. **Their framework doesn't operate on the standard notations as defined by the EAST-ADL2 or the AUTOSAR.** AutoMoDe uses the AutoFocus notation which is very closely related to the subset of the UML 2.0 concepts. **Finally the AutoFocus is not defined as an instance of the architecture framework.**

5.1.3. Automotive Architecture Framework

There is only one work which discusses the first concept of an architecture framework for automotive systems also called Automotive Architecture Framework [93]. The authors of this paper define four levels.

The *meta Architecture Framework* (mAF) is the most generic level. It introduces standard architecture framework concepts (view, viewpoint, interface, component, concern). This level of abstraction can be compared to the definition of architecture frameworks presented in the IEEE 42010 standard. However, it contains some minor differences. It extends the standard specification of frameworks with some additional elements and relations. For instance, mAF specifies *scope*, which contains a *boundary* (a specific type of view) and *sub-scopes*. Also, a view may contain sub-views and descriptions that might be formal or informal. Lastly, concern contains *metrics*.

The *common Architecture Framework* (cAF) defines terms that are relevant for each type of a system. Elements of the cAF are divided into three main parts, which define the layers of abstraction of the system to be built. These parts are described as the following views: *Functional Architecture*, *Logical Architecture* and *Technical Architecture*. The Functional Architecture describes the total system as a black-box. User functions that are part of this level describe the functionality visible to the system's environment. Each user function may be further refined into finer-grained user functions. The Logical Architecture presents the system as a white-box. On this level, a system is decomposed into a number of interacting logical components (which might be further decomposed into logical components) that realize the functionality described by the Functional Architecture. This level might also describe the functionality of individual subsystems, which are part of the full system. The Technical Architecture is implementation oriented. It describes how the system specified by means of logical components can be distributed across hardware elements. It is composed of three parts: These are the *Runtime Model*, the *Hardware Topology* and *Allocation*. The first specifies the behavior. Hardware Topology describes the structure of the hardware platform, while the Allocation view relates the elements of the first two views.

The *domain-specific Architecture Framework* (dAF) focuses on specific types of systems (avionics, automotive, etc.). It provides a common terminology, structure, methods, architecture models, guidance and rules for developing, understanding, representing and comparing domain-specific product architectures to different stakeholders. It also provides an insight for external stakeholders into how a specific product is developed.

The last, *organization-specific Architecture Framework* (oAF) adapts dAF to the specific requirements of a particular Original Equipment Manufacturer (OEM).

Respecting the rules of mAF, cAF and dAF, the authors of [93] defined an instance of such a domain-specific Architecture Framework for the automotive industry, called the Automotive Architecture Framework (AAF). The specification of the AAF contains a set of viewpoints which are divided into two subcategories; *mandatory*, which are independent of specific product strategies, and *optional*, which reflect specific OEM focus. The recommended mandatory viewpoints are the *Functional Viewpoint*, the *Technical Viewpoint*, the *Information Viewpoint*, the *Driver/ Vehicle Operations Viewpoint*, and the *Value Net Viewpoint*. The first one views the vehicle as a set of functions and their logical interactions. The *Technical Viewpoint* looks at the car from the perspective of its physical components (electronic and electrical hardware), its behavior (this also includes physical aspects - thermodynamics, acoustics, vibrations, mechanical deformations), its dependencies and its constraints. The *Information Viewpoint* is a specification of information and data objects used to define and manage a vehicle. This includes the description of mechanisms, protocols and standards that support information transfer between vehicle subsystems. The *Driver/Vehicle Operations Viewpoint* presents the vehicle from the driver's point of view. Therefore it describes interactions, interfaces, interdependencies between vehicle and the driver, together with the surrounding environment. The authors also suggest additional optional viewpoints, most of which relate to non-functional concerns such as safety, security, etc. Nevertheless, they do not provide details of any particular example. As part of the future AAF they also envision the following:

- the specification of modeling methods for architectures and their properties
- a meta-language and meta-models for describing the structure and parts of an architecture description
- pragmatic and methodological facets of architectures, including principles, rules and best practices
- general terminology with precise definitions of the relevant notions to be used to describe, discuss, and evaluate architectures
- methods and approaches to assess and evaluate architectures
- a clear understanding and definition of the function and role of architecture in the development process

Authors of [93] provide an interesting, first concept of an architecture framework for the automotive. They admit that their work needs further continuation. **The specification of the AAF done within this thesis responds to this need by addressing some of the features envisioned in [93] and listed before. Specifically, it provides meta-models (EAST-ADL2, AUTOSAR, SysML, MARTE and optimization profile), modeling methods (UML profile mechanism) and finally additional viewpoints that enable the evaluation of architecture (most importantly timing, analysis and optimization viewpoints) which is missing in the work of [93].**

5.1.4. Related Work Conclusions

The advancements in commercial tooling are rapid and clearly visible. Just two years ago there was no commercial tool that would support the EAST-ADL2 language. Nowadays not only the higher level modeling is evolving but the tool suppliers are also trying to integrate ways for analyzing the system level models. The next expected outcome of this evolution is the integration of the techniques for the optimized deployment process.

Table 5.1 highlights the main properties of the existing frameworks discussed before. As it can be seen none of the commercial tools features the full MAO integration. This doesn't apply to the AutoFocus which combines the modeling with the architecture analysis and optimization. The main difference and hence contribution of the AFfMAO in regards to the AutoFocus is the **(1)** usage of the automotive standards (EAST-ADL2 and AUTOSAR) and UML profile mechanism and **(2)** reference to the Architecture Framework. The last employs novel idea of the **(3)** optimization objectives elicitation through the provision of the optimization viewpoint which requires **(4)** defining a new modeling concepts to express the optimization concerns. Also relevant is the usage of the SysML/MARTE as a pivot language to integrate automotive architecture languages, i.e. the EAST-ADL2 and the AUTOSAR.

Property / **Tool**	Higher Level Modeling	Implementation Level Modeling	System Level Analysis	System Level Optimization	Based on AF	Modeling Languages
SystemDesk		✓				AUTOSAR
DaVinci Developer		✓				AUTOSAR
VSx toolset	✓	✓	✓			EAST-ADL2,

						AUTOSAR
MetaEdit+	✓	✓				EAST-ADL2, AUTOSAR
SystemWeaver	✓	✓	✓			EAST-ADL2, AUTOSAR
PREEVision	✓	✓	✓			DSL
Artop		✓				AUTOSAR
SymtaS			✓	✓		AUTOSAR
AutoFocus	✓	✓	✓	✓		AutoFocus specific notation
Proposed Framework (AFfMAO)	✓	✓	✓	✓	✓	EAST-ADL2, AUTOSAR, SysML, MARTE Optimization Lng

Table 5.1. Features of the Frameworks/Tools for the Automotive Domain

5.2. Automotive Framework for Modeling Analysis and Optimization

The AFfMAO was implemented as an instance of a conceptual Automotive Architecture Framework. This relation is shown on the left side of the Figure 5.1. The AAF itself was constructed following the principles of the Architecture Framework as defined in the ISO 42010 standard [1] and introduced in section 2.3. In that respect, description of the AFfMAO will be done implicitly through the specification of the AAF. In substance the Architecture Framework is a set of conventions, principles and practices for the description of architectures within a specific domain and/or community of stakeholders. Consequently specification of the AAF in the following sections will be comprised of the definition of architecture viewpoints with their related concerns, model kinds and correspondence rules.

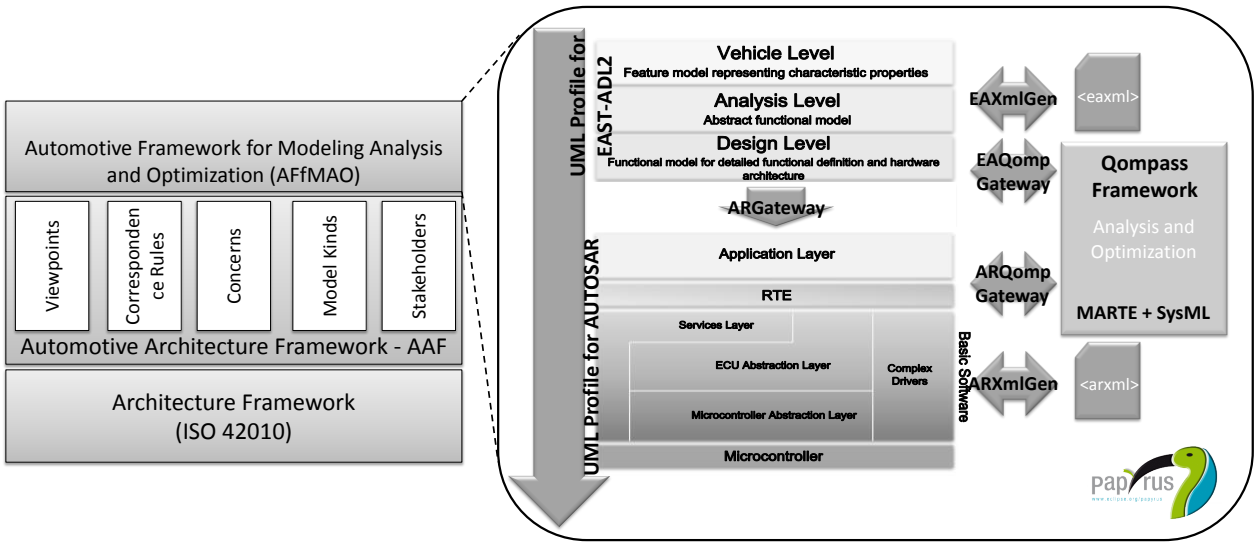

Figure 5.1. AFfMAO built as an instance of the AAF

The right side of the Figure 5.1 presents detailed perspective over the AFfMAO. Relevant information from this figure concerns final choices of the modeling techniques, set of transformations, analysis and optimization engines and platform used to implement the AAF as the AFfMAO.

Platform

The base development platform was Papyrus MDT [94] which is a framework built on top of the Eclipse project [95]. It provides functionality for graphical modeling of UML and SysML languages. Significantly it supports mechanism for constructing UML profiles which is largely used by the AFfMAO.

Modeling Techniques

It can be noticed from the Figure 5.1 that AFfMAO adopts UML profiles for most of the languages. UML profile mechanism is specified as a possible modeling mechanism within the AAF. Therefore, later in this chapter for those languages for which UML profile wasn't defined yet, this work provides its own specification. This refers to the optimization metamodel but also the AUTOSAR.

Analysis and Optimization Engine

The AAF includes viewpoints expressing analysis and optimization concerns (see 5.3.4 and 5.3.5). They are implemented in a separate module called Qompass framework (previously called Optimum [61]). Models created according to the analysis and optimization viewpoints can be analyzed and optimized as Qompass delivers a set of algorithms for timing analysis and optimization of deployment. Algorithms presented in chapter 4 were integrated within the Qompass.

5.3. Viewpoints

The Figure 5.2 presents the viewpoints of the Automotive Architecture Framework. The viewpoint establishes the conventions for constructing, interpreting and analyzing the view to address concerns framed by that viewpoint. The following subsections will detail the characteristics of the proposed viewpoints.

Architecture Modeling Viewpoints

The viewpoints related to the architecture modeling were mostly influenced by the current specification of the EAST-ADL2 and the AUTOSAR although neither of these languages implicitly lists any viewpoint. In fact a viewpoint is not only the inclusive part of the architecture framework but also of the architecture description language such as the EAST-ADL2 or the AUTOSAR. Therefore their absence in their specification might be somewhat surprising. The EAST-ADL2 provides the metamodel presenting the hierarchy of the different models (see Figure 5.3). This inspired the viewpoints of the AAF and their layering. These are the *Feature Analysis Architecture* (relates to *EAST-ADL2::VehicleLevel* model), *Functional Analysis Architecture* (*EAST-ADL2::FeatureAnalysisArchitecture*), *Functional Design Architecture* (*EAST-ADL2::FunctionalDesignArchitecture*), *Hardware Architecture* (*EAST-ADL2::HardwareArchitecture*) and *Allocation* (*EAST-ADL2::Allocation*). Layers specification is not required but it provides more clearance to the AAF description. The EAST-ADL2 doesn't differentiate the models of the Implementation Level. This model corresponds to the Technical Layer therefore viewpoints specification for this layer was done inspecting the main modeling activities included in the AUTOSAR standard. Worth noting is that all these models from the Figure 5.3 combined together represent the system level model. Analogously the presented viewpoints constitute the system level specification.

The design and technical levels contain also timing viewpoint (correspondingly *Timing* and *Application Timing*). It is used to enhance the system model with the additional information related to time. This timing information can be then used by the analysis and optimization viewpoints to configure and evaluate architecture.

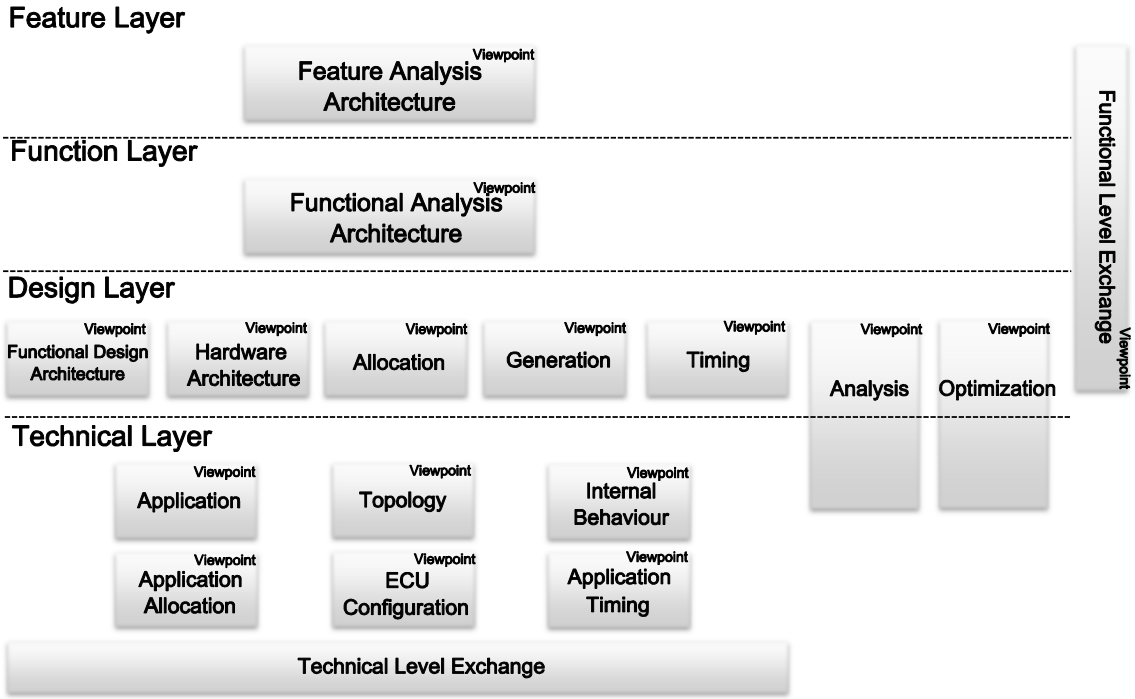

Figure 5.2. Layers and Viewpoints of the Automotive Architecture Framework

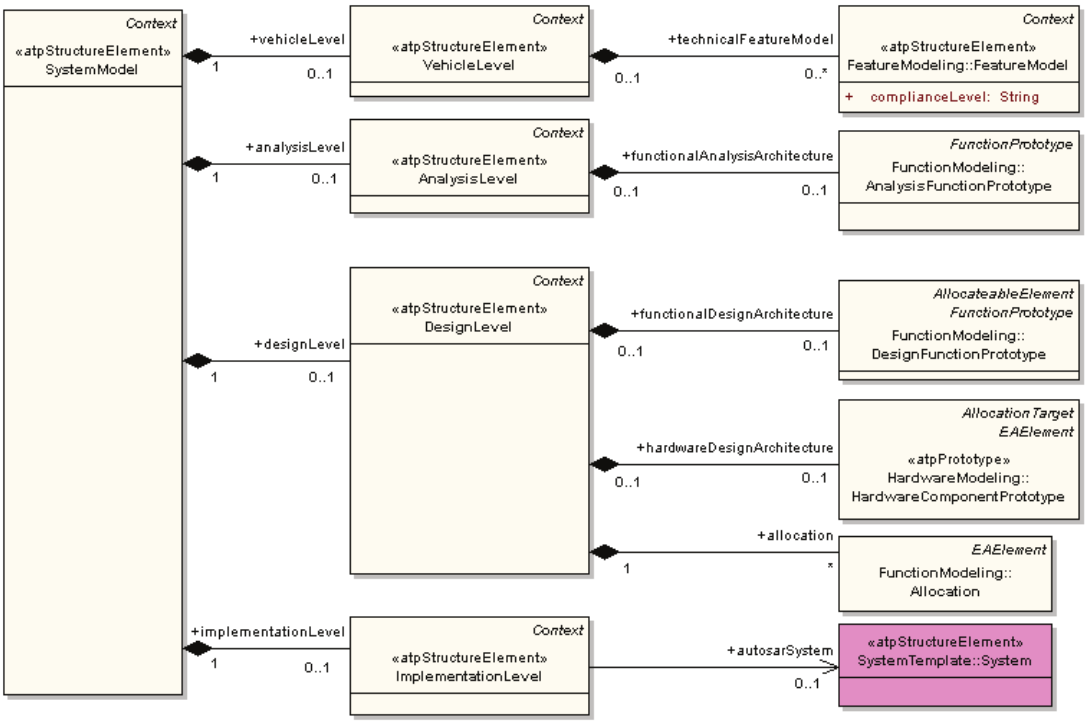

Figure 5.3. Levels of the EAST-ADL2 Model

Analysis and Optimization Viewpoints

The second type of viewpoints relate to the analysis and optimization (*Analysis* Viewpoint and *Optimization* Viewpoint). According to [1] viewpoint conventions can also include the analysis techniques. The analysis viewpoint is an enabler for running the numerous analysis techniques. In the context of this work the analysis techniques relate to the timing and memory overhead analysis. These aspects are also considered by the Optimization Viewpoint which presents the concerns related to the architecture optimization.

Transformation and Exchange Viewpoints

The third type of viewpoints supports the transitions within the framework, i.e. *Generation* Viewpoint and with other frameworks; *Technical Level Exchange* and *Functional Level Exchange* Viewpoints. The first one is concerned about the models expressing the constraints for the generation of the technical layer models out of the design layer models. The next two relate to the standard exchange formats which enable the tools interoperability.

5.3.1. Feature, Functional and Design Level Viewpoints

The Feature, Functional and Design Level are the three levels which abstract from the implementation specific concerns such as the software architecture. The purpose of the feature level is to represent the vehicle as a set of features without particular decision about the way to implement them, i.e. whether in hardware or in software. The functional level should describe the functional composition of a vehicle, its functions, interfaces, interactions, behavior and constraints. It represents the first refinement of the features as defined at the feature level. Following that is the design level in which higher level functions are refined into sub-functions and, finally, into atomic functions which are non-concurrent units. This layer also delivers the specification of hardware resources. The main characteristic of this layer is that it abstracts from a specific platform such as the AUTOSAR.

Feature Analysis Architecture Viewpoint (FAA)

The *Feature Analysis Architecture* Viewpoint (FAA) is located at the feature layer. The concern of this viewpoint is to specify the features of a vehicle, links between them and data types. This viewpoint aggregates one model kind which defines conventions for a type of modeling. According to [1] an architecture viewpoint for each identified model kind shall specify the languages, notations, conventions, modeling techniques, analytical methods and/or other operations to be used on the models of this kind.

The language used to express the concerns of the FAA is EAST-ADL2. The Figure 5.4 is a snapshot of the EAST-ADL2 [6] metamodel used by the FAA. The modeling technique relates to the UML profile mechanism. The UML profile itself comes from [96]. Diagram that is used to model the features with the profile is the UML Composite diagram. All the viewpoints from the feature, function and design layer except the analysis, optimization and functional level exchange, uses dedicated parts of the EAST-ADL2 metamodel from [6] and UML profile as defined in [96].

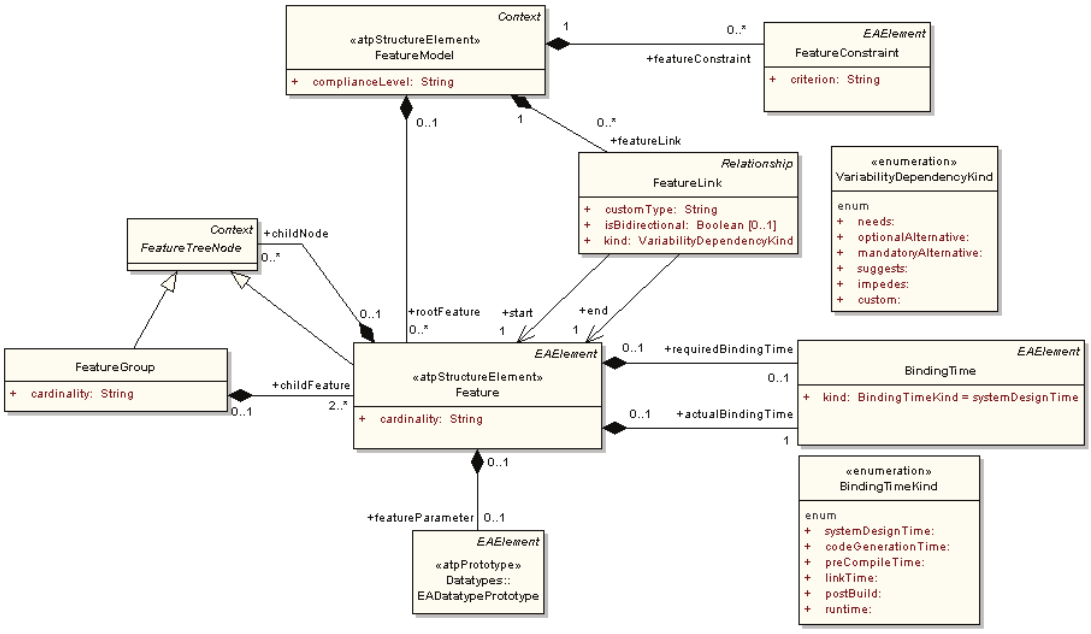

Figure 5.4. Part of the EAST-ADL2 Metamodel for the FAA from [6]

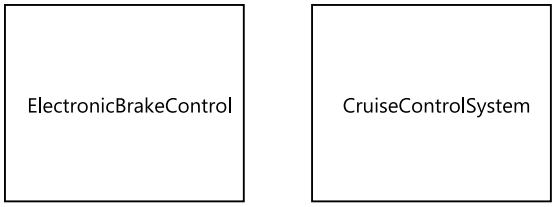

Figure 5.5. Model of two Features

Functional Analysis Architecture Viewpoint (FunAA)

The *Functional Analysis Architecture Viewpoint (FunAA)* is located at the function layer. The concern of this viewpoint is to specify the system functions which provide a solution to the features modeled at the FAA viewpoint. This viewpoint has two model kinds. The first one called *Functions Types* serves to specify the types of the functions located at the functional layer. The Figure 5.6 is a fragment of the EAST-ADL2 metamodel whose part related to the *AnalysisFunctionType* is used by this model kind. The diagram that it uses this is the UML Composite diagram. The second model kind called *Functions Prototypes* serves to define the instances of the types modeled under the first model kind. The metamodel used refers to the *AnalysisFunctionPrototype* element from the Figure 5.6. These elements are modeled within the UML Composite diagram.

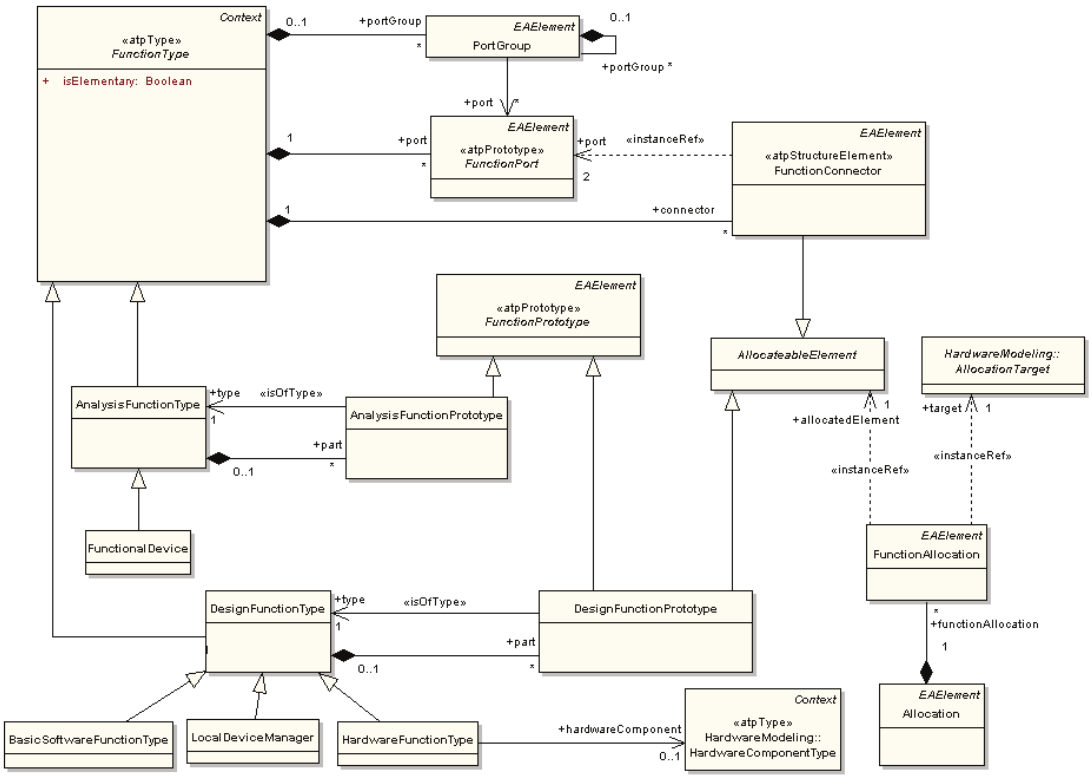

Figure 5.6. Part of the EAST-ADL2 Metamodel for the FunAA and FDA from [6]

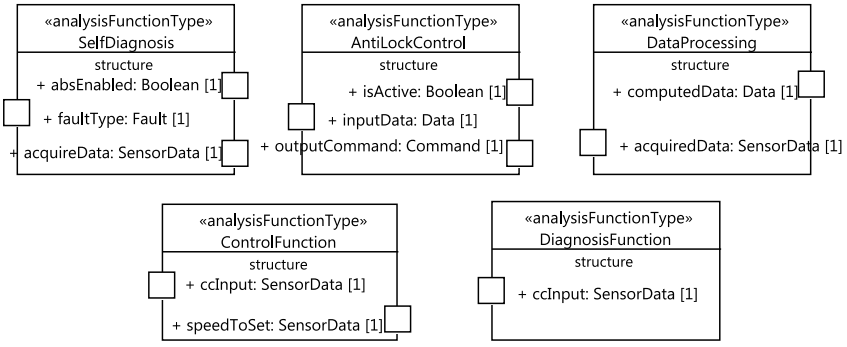

Figure 5.7. Function Types at the Function Layer

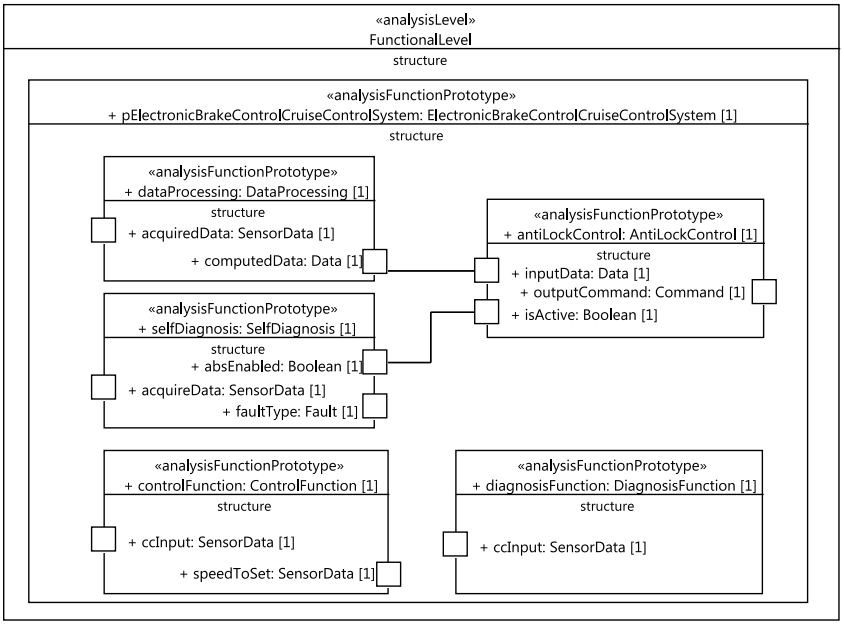

Figure 5.8. Function Prototypes at the Function Layer

Functional Design Architecture Viewpoint (FDA)

The *Functional Design Architecture* Viewpoint (FDA) is located at the design layer. The concern framed by this viewpoint refers to the further refinement of the functions specified at the functional layer. There are two model kinds. First called *Design Functions Types* serves to model the function types at the design layer. It uses the UML Composite diagram and for the metamodel it is the one from the Figure 5.6 related to the *DesignFunctionType* element. The second model kind, *Design Function Prototype* as its metamodel uses the specification related to the *DesignFunctionPrototype* element from the Figure 5.6. The function prototypes are modeled using the UML Composite diagram.

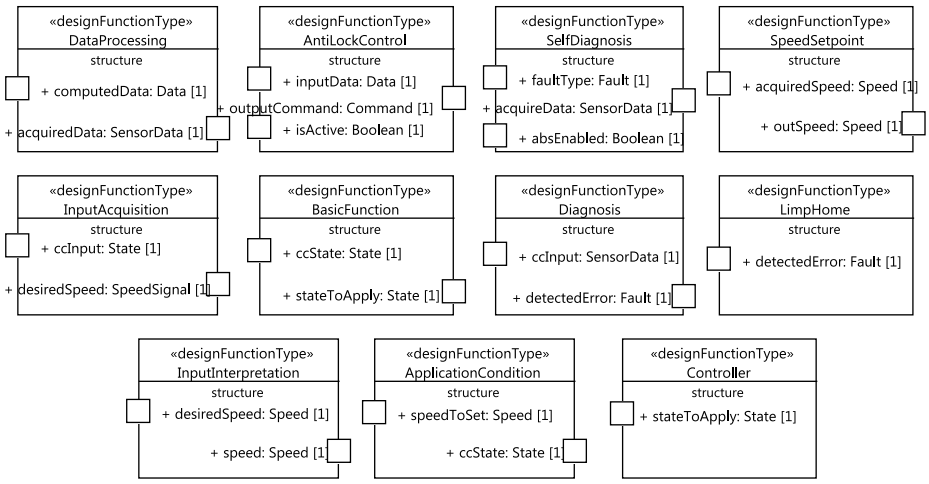

Figure 5.9. Function Types at the Design Layer

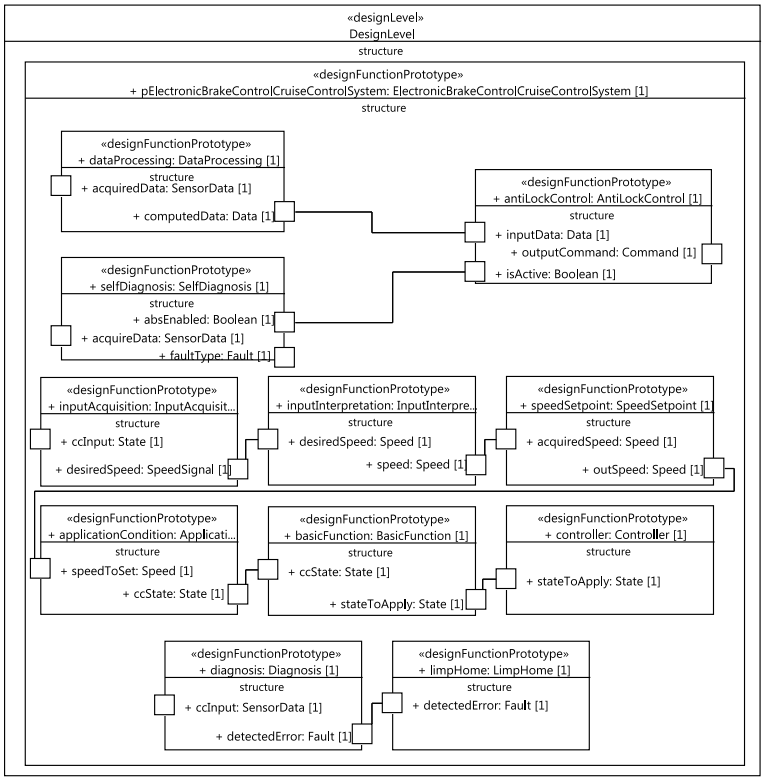

Figure 5.10. Function Prototypes at the Design Layer

Hardware Architecture Viewpoint (HA)

The *Hardware Architecture* Viewpoint (HA) is also part of the design layer. Its concern is provision of an abstract hardware platform specification. This relates to the sensors, actuators, ECUs and their topology defined through the direct links and the communication BUSes (see Figure 5.11). There are two model kinds for this viewpoint. First called *Hardware Types* uses the

UML Composite diagram and the part of the metamodel from the Figure 5.11 related to the *HardwareComponentType*. The second model kind called *Hardware Prototypes* uses the UML Composite diagram and the part of the metamodel from the Figure 5.11 related to the *HardwareComponentPrototype*.

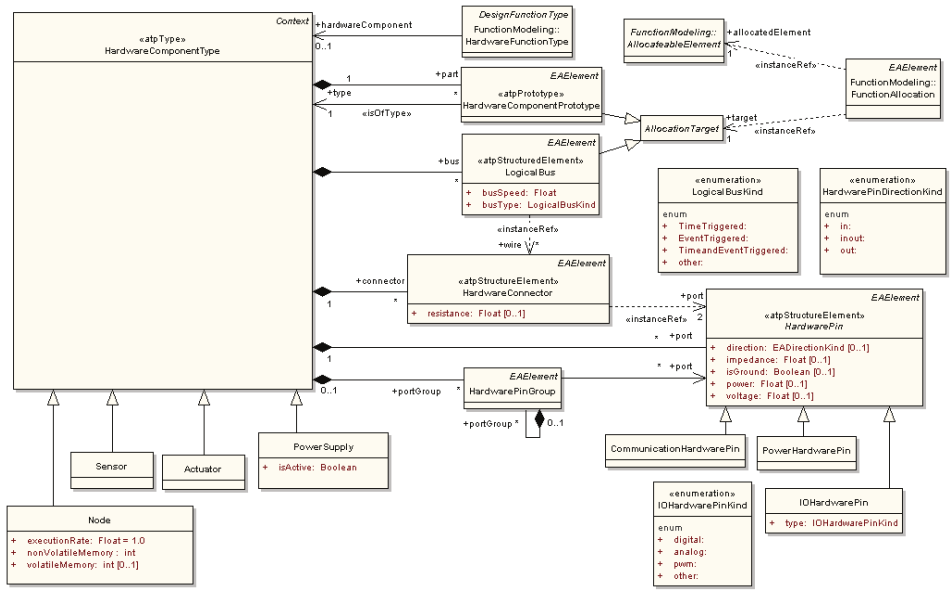

Figure 5.11. Hardware Architecture Modeling in the EAST-ADL2

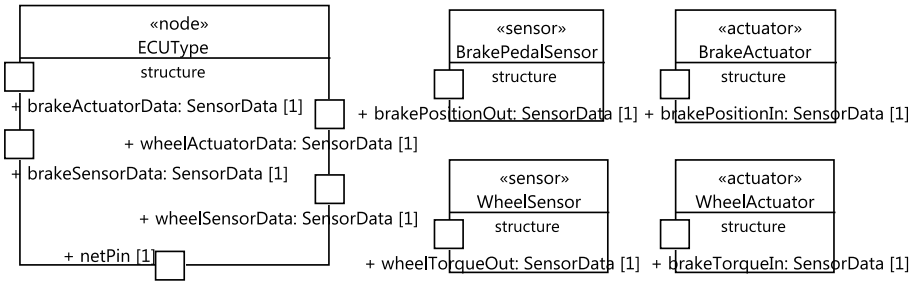

Figure 5.12. Model of the Hardware Types

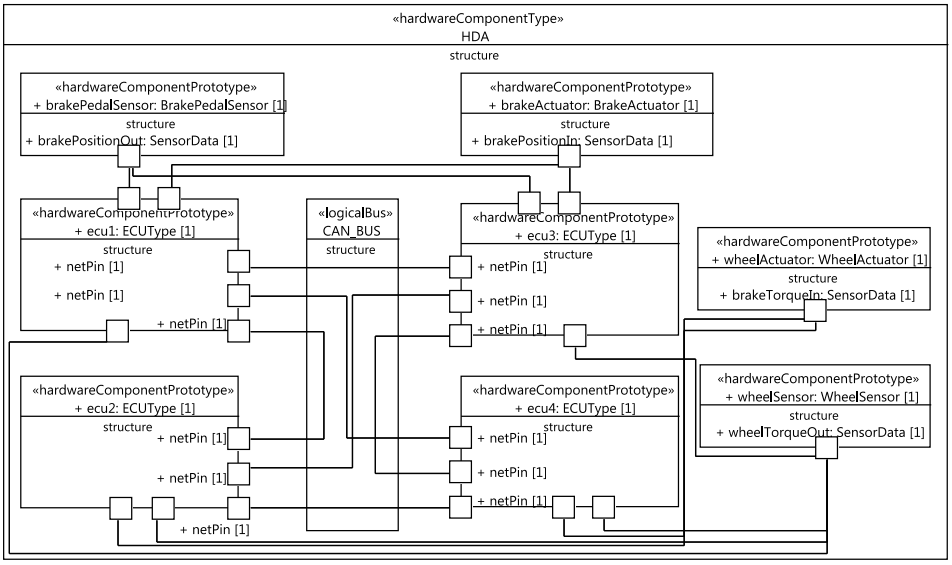

Figure 5.13. Model of Hardware Prototypes

Allocation Viewpoint

The *Allocation* Viewpoint from the design layer relates the elements of the FDA and the HA viewpoints. The only element used is a stereotype called *Allocation* applied on the UML Dependency (see the Figure 5.6). Diagram used to model the allocation concerns this is the UML Composite diagram.

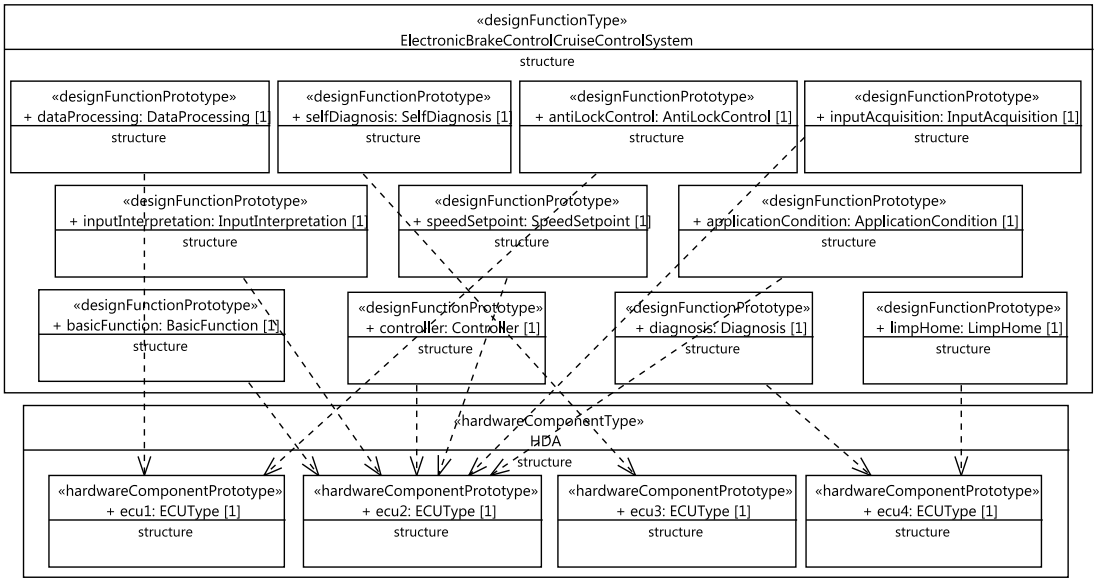

Figure 5.14. Model of the Allocation

Timing Viewpoint

A key concern when developing automotive systems is to take into account non-functional requirements. These can include the timing constraints imposed on the functions, safety, and limitations on memory usage and power consumption. For this reason an AAF should include viewpoints that refer to non-functional concerns. Besides as it was already shown in the context of the optimization techniques, non-functional requirements such as end-to-end deadlines can drive the design. To meet these expectations the *Timing* Viewpoint is included into the AAF specification. Its concern is to provide the timing characteristics such as events' periods, functions' execution times and the timing constraints such as end-to-end deadlines, synchronization constraints or time budgets. The EAST-ADL2 metamodel contains the elements which enable to enhance the models with additional information related to time. Consequently within this viewpoint designer can model the timing properties of each function or specify the end-to-end flows for which the deadlines can be assigned, etc. This viewpoint is attached to the viewpoints describing the static architecture, i.e. FAA, FunAA and FDA as the timing description needs to reference the system architecture.

The information provided under this viewpoint serves to run early stage timing analysis such as the computation of the utilization for each ECU/BUS if of course the allocation is specified. The timing model is used during the generation of the analysis model governed by the analysis viewpoint (see section 5.3.3) if the analysis itself concerns timing analysis such as the utilization computation.

The Figure 5.15 is an example of the model specified under the timing viewpoint. The timing information present in this model is a subset of the overall timing specification related to the CCS & ABS use case. It shows specification of one end-to-end flow (event chain), its triggering event, period and deadline. There is also information about the execution time of the *dataProcessing* atomic function.

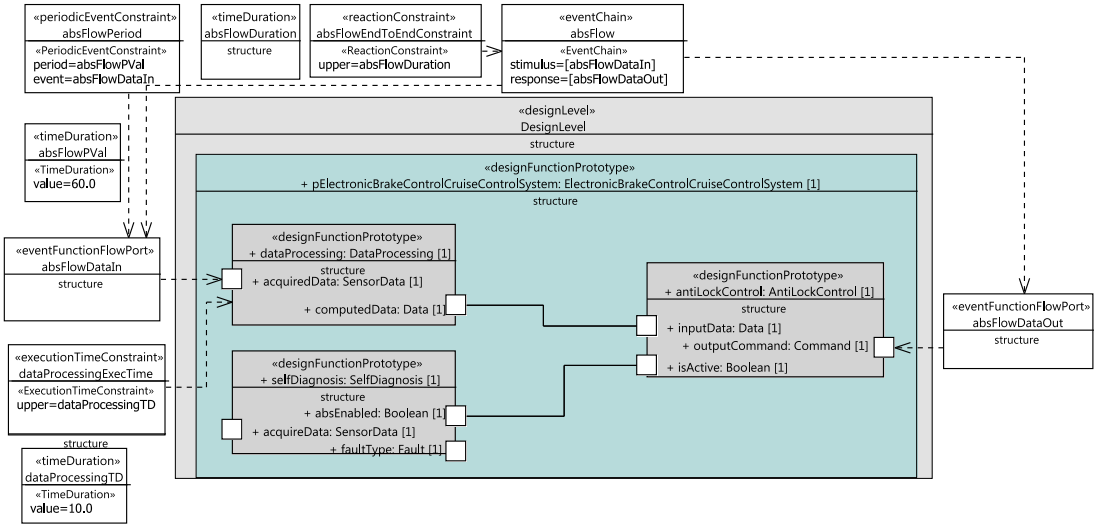

Figure 5.15. Model with Timing Information

5.3.2. Technical Level Viewpoints

Technical level viewpoints refer to the AUTOSAR standard. They use the AUTOSAR metamodel to express their concerns which in general relate to software and hardware architecture specification.

Application Viewpoint

The concern of the Application Viewpoint is the specification of the software architecture. Its fundamental part is the specification of software components, ports, interfaces and data elements. For the modeling of software entities *SwComponentPrototype* is used, typed with *SwComponentType*. There are few types of *SwComponentType* where the most significant are *AtomicSwComponentType* and *CompositionSwComponentType*. The last is to allow encapsulation of specific functionality by aggregating existing software components. Since it inherits from the *SwComponentType* it can be aggregated as well. This is solely an architectural element and serves only to take away the complexity when viewing or designing logical software architecture.

For the purpose of communication, software components can have ports (*PortPrototype*) which are characterized by interfaces (*PortInterface*). The Table 5.2 presents the UML profile used for this viewpoint by showing the UML extensions for the key elements used to specify the application. There are two model kinds using this UML profile. These are the *Software Types* and *Software Prototypes* model kind. The first one uses the UML Class diagram to define the types

for the software components. Second one is used for the specification of software components instances using the UML Composite diagram.

The Figure 5.16 and Figure 5.17 present the model of software component types and prototypes. This entire model in order to be complete should also contain a UML Property stereotyped with *SwComponentPrototype* and typed with *ElectronicBrakeControlCruiseControlSystem*.

AUTOSAR Concept	**UML Extension**
SwComponentType	Class
SwComponentPrototype	Property
PortPrototype	Port
PortInterface	Interface

Table 5.2. UML Profile for the AUTOSAR metamodel used by the Application Viewpoint

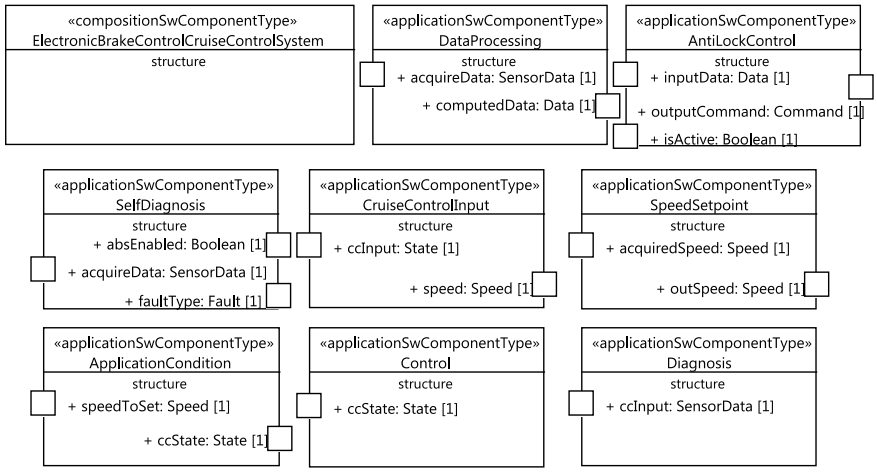

Figure 5.16. Model of Software Component Types

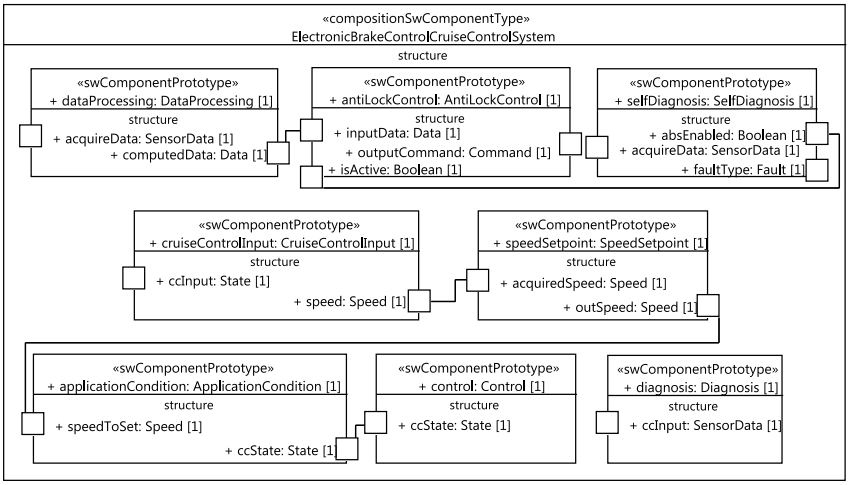

Figure 5.17. Model of Software Component Prototypes

Topology Viewpoint

The *Topology* Viewpoint has the same purpose as the Hardware Architecture viewpoint from the design layer. The difference lies in the language used to express its concerns. Analogously as in the case of the previous viewpoints, this one aggregates two model kinds. The first one is called *Topology Types* to specify the types of the hardware elements. The second *Topology Prototypes* delivers the instances of the previously defined types. Both of these model kinds use the UML Composite diagram and the UML profile shown in the Table 5.3.

AUTOSAR Concept	**UML Extension**
ECU	Class
SensorHw	Class
ActuatorHw	Class
ECUInstance	Property
PhysicalChannel	Connector
CommunicationConnector	Port
CommunicationController	Property
CommunicationCluster	Class
HwPort	Port

Table 5.3. UML Profile for the AUTOSAR metamodel used by the Topology Viewpoint

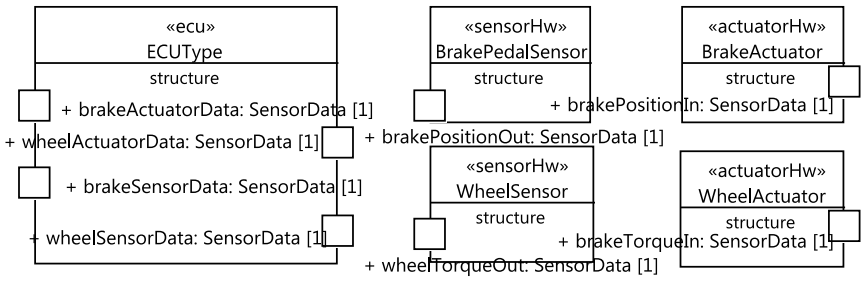

Figure 5.18. Model of Hardware Types based on the AUTOSAR Standard

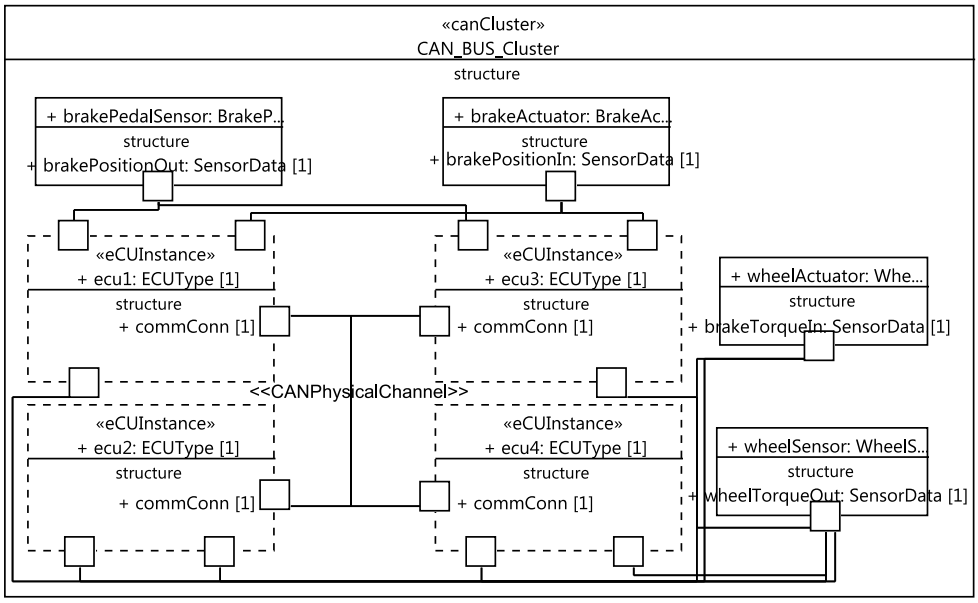

Figure 5.19. Model of Hardware Prototypes based on the AUTOSAR Standard

Internal Behavior Viewpoint

The *Internal Behavior* Viewpoint serves to describe the behavioral decomposition of the software components. The internal behavior describes the scheduling relevant aspects of a component, i.e. the runnable entities (*RunnableEntity*) and the events (*RTEEvent*). Furthermore the behavior specifies which runnable responds to which event. There is only one model kind aggregated by this viewpoint. The UML profile that it uses presents the Table 5.4. The diagram used to model concerns of this viewpoint is the UML Activity diagram.

AUTOSAR Concept	**UML Extension**
InternalBehavior	Activity

RunnableEntity	Activity
RTEEvent	Event

Table 5.4. UML Profile for the AUTOSAR metamodel used by the Internal Behavior Viewpoint

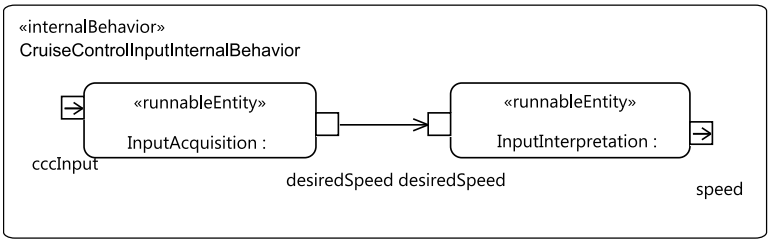

Figure 5.20. Model of the Internal Behavior for the CruiseControlInput Software Component

Application Allocation Viewpoint

The *Allocation* Viewpoint from the technical layer relates the elements of the Application viewpoint with the elements of the Topology viewpoint. The software components are allocated on the ECUs using the *SwcToEcuMapping* meta-element. It holds a reference to the ECU (ECUInstance) and a reference to all of those software component instances that will be allocated on this particular ECU. Hence the concern of this viewpoint is the partial specification of the deployment. It is called partial as this viewpoint doesn't hold information about runnables partitioning and tasks scheduling. The Table 5.5 presents the UML extension for the *SwcToEcuMapping* artifact. Diagram used by the Allocation viewpoint this is the UML Composite diagram. The model of allocation specified under this viewpoint is analogous to what is shown on the Figure 5.14, except that here the dependencies are stereotyped with the *SwcToEcuMapping* stereotype, the allocated entities these are software component prototypes *SwcComponentPrototype* and allocation target this is *ECUInstance.*

AUTOSAR Concept	**UML Extension**
SwcToEcuMapping	Constraint

Table 5.5. UML Profile for the AUTOSAR metamodel used by the Allocation Viewpoint

ECU Configuration Viewpoint

This viewpoint corresponds to the phase of the AUTOSAR methodology called ECU configuration [97]. One of the configuration procedures is the partitioning of the runnables in OS tasks and assignment of priorities for them. Therefore the concern of this viewpoint is to complete the deployment specification initiated by the Application Allocation viewpoint by formulating the tasks, specifying runnables partitioning and priorities assignment.

The configuration language of AUTOSAR uses containers and actual parameters. Containers are used to group corresponding parameters. Parameters hold the relevant value that configures the specific parts of an ECU. The process of ECU configuration is twofold. First is specification of *ECU Configuration Parameter Definition* and the second is specification of *ECU Configuration Value*. *ECU Configuration Definition* declares how and what information will be presented whereas *ECU Configuration Value* holds the actual configuration. For the first part, there are existing AUTOSAR templates that define standard configuration descriptions, e.g. how to proceed with the definition of OS specification. However vendor might want to specify his own, specific ECU configuration parameters. This can be achieved thanks to the flexibility delivered by the meta-model for ECU configuration (see Figure 5.21).

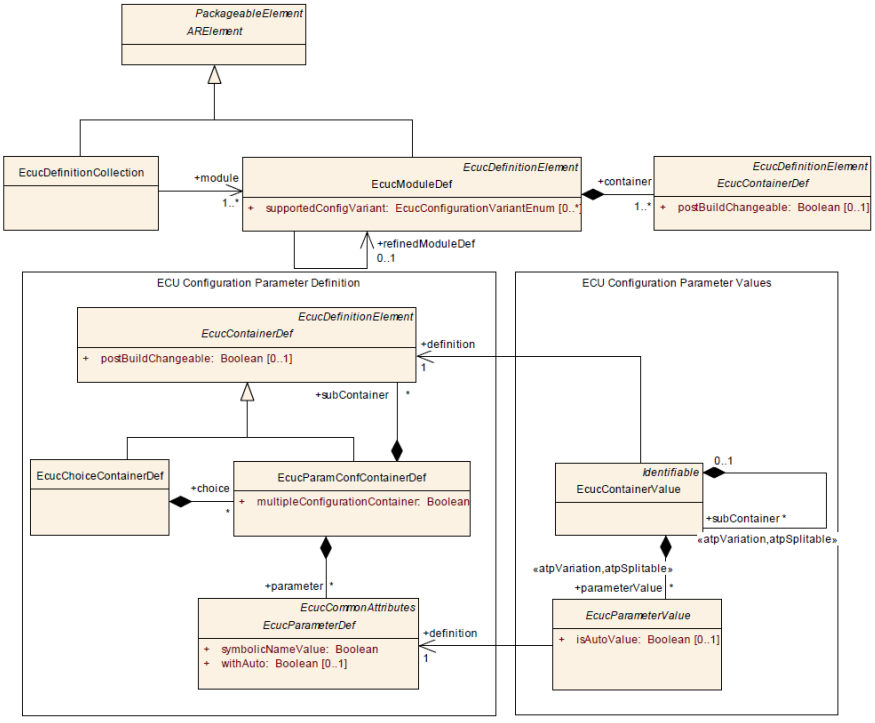

Figure 5.21. Metamodel used for the ECU Configuration [97]

Consequently specification of the tasks for each ECU is also a twofold process. In this case we are following the standard template as defined by the AUTOSAR.

1. Definition of parameters
    a. Creating an instance of *EcucModuleDef* named “OS” (Operating System)
    b. Creating an instance of *EcucParamConfContainerDef* named “OsTask” and setting it as a value of property *container* of the previously created “OS”
2. Definition of parameter values, i.e.
    a. Creating an instance of *EcucModuleConfigurationValues* with its property *definition* set to the element created in 1.a, i.e. “OS”.
    b. Representing each task as an instance of *EcucContainerValue* and setting its property *definition* to the element created in 1.b, i.e. “OsTask”.

Partitioning of runnables into OS tasks is done indirectly. Tasks are correlated with events (*RTEEvent*) that trigger runnable entities. This modeling is also a two steps procedure.

1. Definition of parameters
    a. Creating an instance of *EcucModuleDef* named “Rte”
    b. Creating container, i.e. an instance of *EcucParamConfContainerDef* named “RteSwComponentInstance” and setting it as a value of property *container* of the previously created “Rte”
    c. Creating another container, i.e. an instance of *EcucParamConfContainerDef* named “RteEventToTaskMapping” and setting it as a value of property *subContainer* of the previoiusly created “RteSwComponentInstance”. This one allows referencing previously specified OS tasks and *RTEEvents*.
2. Definition of parameter values
    a. Creating an instance of *EcucModuleConfigurationValues* with property *definition* set to element created in 1.a, i.e. “Rte”
    b. Creating an instance of *EcucContainerValue* with property *definition* set to element created in 1.b., i.e. “RteSwComponentInstance”.
    c. Creating an instance of *EcucContainerValue* per each runnable to task mapping, with property *definition* set to element created in 1.c., i.e. “RteEventToTaskMapping”.

Table 5.6 lists AUTOSAR concepts needed by this viewpoint to specify the configuration of an ECU in terms of its OS tasks and partitioning of runnables, together with UML elements that they extend. The Figure 5.22 shows an example of mapping runnable *inputAcquisition* to task *t1*.

	AUTOSAR Concept	UML Extension
ECU Configuration Parameter Definition	EcucModuleDef	Class
	EcucParamConfContainerDef	Class
	EcucReferenceDef	Property
	EcucForeignReferenceDef	Property
ECU Configuration Values	EcucModuleConfigurationValues	InstanceSpecification
	EcucContainerValue	InstanceSpecification
	EcucReferenceValue	InstanceValue
	EcucInstanceReferenceValue	InstanceValue

Table 5.6. UML Profile for the AUTOSAR metamodel used by the ECU Configuration Viewpoint

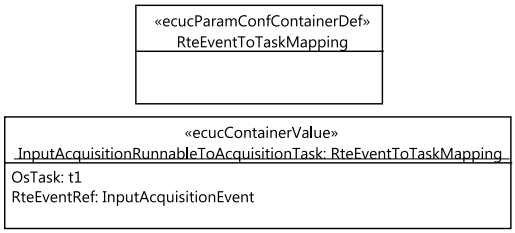

Figure 5.22. Partitioning of the Runnable InputAcquisition in the task t1

Application Timing Viewpoint

This viewpoint has the same concern as the Timing viewpoint defined for the higher abstraction levels. The main difference lies in the language concepts defined for the AUTOSAR timing extension [8]. Most of them reflect the elements from the EAST-ADL2 timing metamodel using even the same names in many cases. The fundamental notion for the description of timing properties is the notion of event chain, specified through the *TimingDescriptionEventChain* element. A timing event chain expresses the temporal correlation between two observable timing events, namely *stimulus* and *response* that have functional dependency. Timing events are specified through so-called *TimingDescriptionEvent* elements. Event chains can be built from sub event-chains (segments). Triggering behavior (e.g. periodic, sporadic, and arbitrary) of event

chains are specified through *EventTriggeringConstraint* element that refer to the stimulus of the corresponding event chain. An event chain is used as the subject to attach a timing constraint, represented by *LatencyTimingConstraint* elements. Actually, event chains can be defined at different levels of granularity, in accordance with the *Timing View* concept of AUTOSAR. First level called *VfbTiming* deals with timing information related to the interaction of software components at the VFB (Virtual Function Bus) level. Next, *SwcTiming* extends timing specification with timing properties of software component's internal behavior. For example at this level it is possible to specify the WCETs of runnable entities. The *SystemTiming* includes information about topology, software deployment, and signals mapping in timing specification. The *BswModuleTiming* focuses on the activation, start and end of the execution of basic software module entities. Finally, at the *EcuTiming* level, timing can reference all the ECU-relevant information, e.g. an ECU bus communication. This work is interested in the *SystemTiming* level. This level of timing information refinement fits to the level of information required by the timing analysis techniques and optimization algorithms as employed for the analysis and optimization viewpoints. Possible advancement of the AAF with other techniques requiring more refined timing information would naturally lead to the inclusion of other timing description levels. This viewpoint analogously to the timing viewpoint at the EAST-ADL2 level references the elements modeled under the internal behavior and application viewpoints. The diagram used to express the concerns of this viewpoint this is the UML Composite diagram. The key modeling elements and its corresponding UML profile are specified in the Table 5.7.

AUTOSAR Concept	**UML Extension**
TimingExtension	Comment
TimingConstraint	Constraint
TimingDescriptionEvent	TimeObservation
TimingDescriptionEventChain	InformationFlow

Table 5.7. UML Profile for the AUTOSAR Timing Extension Metamodel used by the Application Timing Viewpoint

The two below figures are the models expressing the timing concerns. The model from the Figure 5.23 specifies the end-to-end flow between the events occurring on the InputAcquisition and the Controller runnables (SensorDataAcquired and ThrottleTorqueComputed events), with

corresponding end-to-end deadline of 40ms. The Figure 5.24 represents the model which adds the information about the period of the previously defined end-to-end flow. This is done implicitly by assigning a period value to the event of the sink runnable, i.e. SensorDataAcquired. Analogous models should be defined to include the rest of the necessary timing information, i.e. deadline of the remaining three end-to-end flows and their periods.

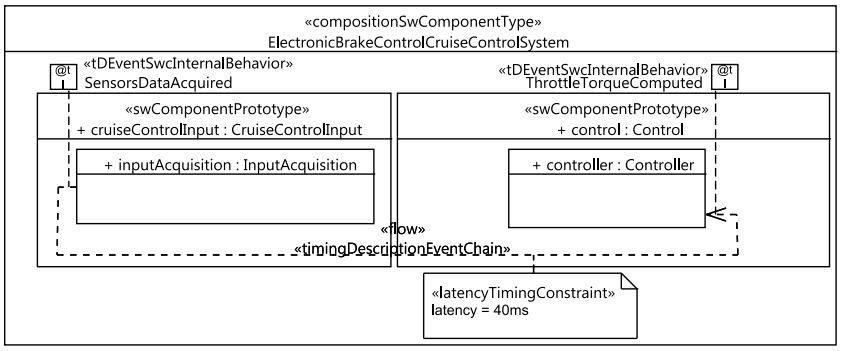

Figure 5.23. Specification of a Latency Constraint for the End-to-End flow under the Application Timing Viewpoint (at the AUTOSAR SystemTiming Level)

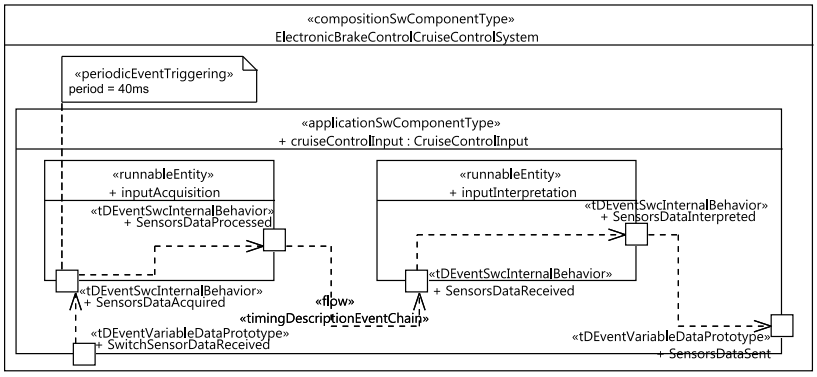

Figure 5.24. Specification of the Activation Period under the Application Timing Viewpoint (at the AUTOSAR SystemTiming Level)

5.3.3. Generation Viewpoint

The generation viewpoint expresses the concern related to the generation of the preliminary implementation model based on the AUTOSAR out of the functional model based on the EAST-ADL2. The necessity of this viewpoint is a consequence of the multiple possible strategies to consider when generating the software architecture. For example one possibility is to generate

one runnable entity from one atomic function and embed each of these runnables in one software component as in [98]. However more advanced schemes should be also allowed, e.g. generation of one runnable out of more than one atomic function. Generation viewpoint with established modeling rules can be a base for the implementation of the generators that will produce the AUTOSAR model out of the EAST-ADL2 model.

The strategies that are considered in this work concern:

- Generation of the runnables out of the atomic functions. There are three possibilities:
  - 1-to-1 – there will be one runnable entity generated out of one atomic function
  - n-to-1 – one runnable entity will be generated out of n atomic functions
  - customized generation
- Generation of the software components:
  - one software component per one runnable entity
  - customized composition of runnables within software components

The possibility to express different strategies for the generation of software architecture is granted by the dedicated metamodel developed for this purpose and its corresponding UML profile. The metamodel comprise of four artifacts presented and described in the Table 5.8. The UML profile called Generation Strategy Profile (GSP) is shown on the Figure 5.25.

Generation Profile Concept	**Semantics**
RunnableGeneration	Serves to specify a way in which runnables will be generated from elementary design functions. The property *sourceFunction* is a list of functions from which single runnable will be generated. Second property, *nameOfGeneratedRunnable* allows specifying the name of a runnable to be generated.
SwcGeneration	Is an abstract stereotype extended by two types of software components that can be generated, i.e. atomic and composite software component. It contains property *nameOfGeneratedSWC* to specify the name of a swc component that will be generated.
AtomicSwcGeneration	Serves to specify a way to aggregate runnable entities that will be generated, into the atomic software components. The property *runnablesToCompose* is a list of those runnable entities that will be aggregated to the same software component.
CompositeSwcGeneration	Serves to specify a way to aggregate generated software components

	into the composite software components. Its property *containedSwc* is a list of those software components (either atomic or another composite) that will be aggregated together.

Table 5.8. Metamodel for the specification of Generation Strategy

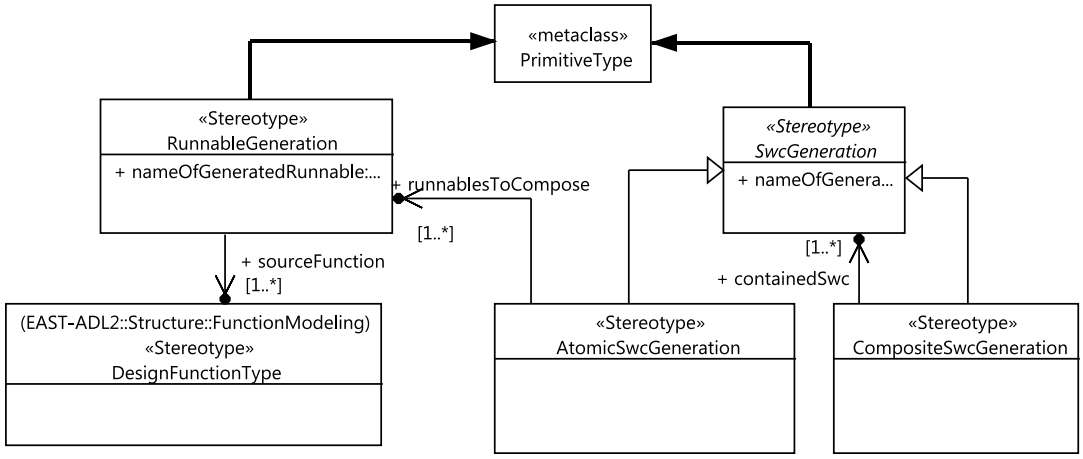

Figure 5.25. UML Profile used to specify the Generation Strategy within the Generation Viewpoint

The Figure 5.26 shows the model with the generation strategy for the Cruise Control & Anti-lock Braking use case. In order to unburden the designer from the specification of the entire generation strategy, the generator itself might employ some predefined scenarios. For example in case if an implicit specification of the generation strategy for particular design function is absent, the generation will produce one runnable entity out of this function, i.e. one-to-one strategy will be applied. Analogously can be done for the mapping of runnables in software components. Possible approach is to assume that for each runnable without predefined mapping in software components, it will be always mapped in a software component containing only this runnable. This approach would simplify the generation model as is the case for the Figure 5.26. In the model from this figure, the instances of *RunnableGeneration* are created only for those design functions which corresponding runnable entities will be ultimately sharing software component with other runnable. For the rest of the design functions, one corresponding runnable will be generated which then will be aggregated in its unique atomic software component. This strategy

was employed by the ARGateway which is a generation plugin embedded in the AFfMAO (see Appendix A for more details on the ARGateway).

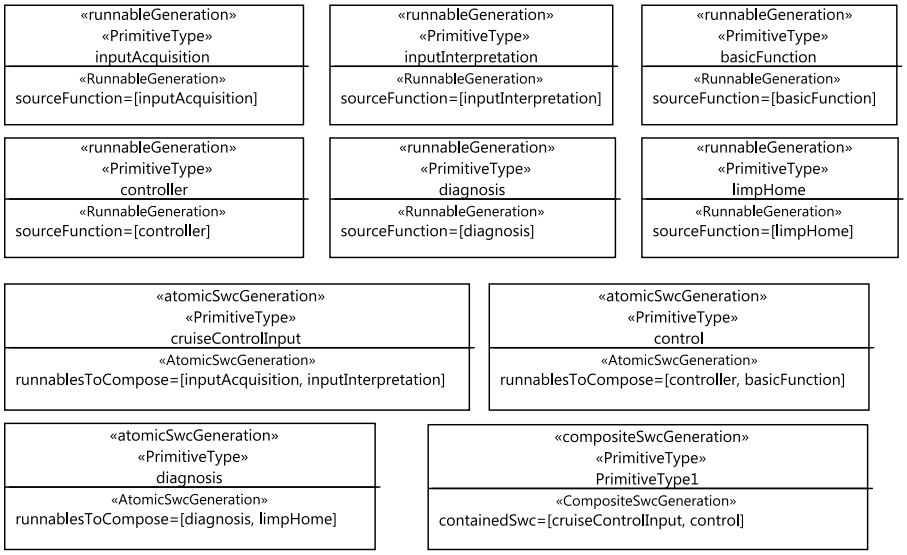

Figure 5.26. Generation Model providing a strategy for the generation of the Runnable Entities and the Software Components

5.3.4. Analysis Viewpoint

This viewpoint main concern is to enable the analysis of automotive architectures. The main criteria according to which it is analyzed are the timing properties, such as the observance of the end-to-end deadlines or the utilization constraints specified for each ECU/BUS. Additional concern relates to the analysis of the memory consumption. Therefore the analysis that can be currently run on the models governed by this viewpoint are the schedulability analysis test, computation of resources utilization and memory consumption. Of course there might be other analysis concerns such as the dependability analysis, etc.

This viewpoint uses a subset of the MARTE and the SysML language. The MARTE subset comes in a form of few packages; GQAM (Generic Quantitative Analysis Modeling), GRM (Generic Resource Modeling), SAM (Schedulability Analysis Modeling) and HRM (Hardware Resource Modeling). The SAM uses similar domain concepts as the GQAM extending it with few artifacts which are specific to the theory of schedulability analysis. An example is the concept of end-to-end flow (SAM::SaEndToEndFlow) which is not present in the GQAM.

The base for this viewpoint is the UML4SysML activity diagram for modeling the control flows and events. The elements of this diagram are extended in a specification of a profile for the GQAM and SAM. This way they can be stereotyped with the MARTE to enrich the model with non-functional concerns and properties used to analyze the architecture. Additionally this viewpoint uses the SysML Block and the Internal Block diagrams to model the platform resources such as sensors, computing resources or shared resources. The SysML elements of these diagrams are enriched with the MARTE stereotypes relating to the platform resources which come from the GQAM, GRM and HRM package.

The central part of the GQAM is called analysis context. An analysis context is the root concept used to collect the information relevant for analysis scenario. It was formalized in [99]. In principle according to the Definition 5.1, the analysis context consists of the specification of end-to-end flows, resources platform and specification of a deployment, i.e. allocation, partitioning, scheduling and ordering (if time driven model is considered). The Table 5.9 below specifies the subset of the MARTE elements used by this viewpoint and the SysML elements that they extend.

The designer might construct multiple analysis contexts, one per each analysis scenario. Their presence in the architecture definition can serve to trace type of analysis performed during the system construction.

Definition 5.1 – Analysis Context: *The analysis context is defined as a triple* $\boldsymbol{\Lambda} = (\boldsymbol{\Gamma}, \boldsymbol{\chi}, \boldsymbol{\partial})$ *where* $\boldsymbol{\Gamma} = \{\boldsymbol{\Gamma_1}, \boldsymbol{\Gamma_2}, \dots, \boldsymbol{\Gamma_n}\}$ *represents a workload behavior, i.e. set of transactions/paths (depending on activation model),* $\boldsymbol{\chi} = \{\boldsymbol{\chi_1}, \boldsymbol{\chi_2}, \dots, \boldsymbol{\chi_n}\}$ *represents resources platform, i.e. set of processing resources and* ∂ *is a deployment specification.*

MARTE concept	**SysML base concept**
GQAM::WorkloadBehavior	UML4SysML::Activity
GQAM::GaResourcesPlatform	SysML::Block
GQAM::GaWorkloadEvent	UML4SysML::AcceptEventAction
GQAM::GaAnalysisContext	UML4SysML::Package
GRM::SchedulableResource	SysML::Part
HRM::HwComputingResource	SysML::Block
HRM::HwBus	SysML::Block
HRM::HwEndPoint	SysML::FlowPort

HRM::HwActuator	SysML::Block
HRM::HwSensor	SysML::Block
SAM::SaExecHost	SysML::Block
SAM::SaCommHost	SysML::Block
SAM::SaStep	UML4SysML::CallAction
SAM::SaSharedResource	SysML::Block
SAM::SaAnalysisContext	UML4SysML::Package
SAM::SaEndToEndFlow	UML4SysML::ActivityPartition
SAM::SaCommStep	UML4SysML::ObjectFlow

Table 5.9. MARTE subset used for the Analysis Context and its SysML Extensions

This work in addition to the analysis context provides also the notion of the *partial analysis context*. In principle according to the Definition 5.2 it is the analysis context but without the deployment specification. The partial analysis context describes the input for the optimization techniques whereas analysis context might be a product of the optimization but also manual configuration.

Definition 5.2 – Partial Analysis Context: *The partial analysis context is defined as a tuple* $\boldsymbol{\Lambda^I = (\Gamma, \chi)}$ *where* $\boldsymbol{\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}}$ *represents a workload behavior, i.e. set of transactions/paths (depending on activation model) and* $\boldsymbol{\chi = \{\chi_1, \chi_2, \dots, \chi_n\}}$ *represents resources platform, i.e. set of processing resources.*

The Figure 5.27 is a model created within the analysis viewpoint. It represents the workload specification, i.e. $\boldsymbol{\Gamma}$. There are four end-to-end flows, i.e. absFlow, selfDiagnosisFlow, controlFlow and diagnosisFlow. Each end-to-end flow is represented with the ActivityPartition stereotyped with the SaEndToEndFlow stereotype. The functional entities are represented with the CallBehaviorAction stereotyped with the SaStep. The important property of this stereotype is called *hostDemand* and represents the WCET of the functional entity. The signals are represented with the control flows stereotyped with the SaCommStep. The last also contains property *hostDemand* and hence information about the WCTT (Worst Case Transmission Time) can be provided. The AcceptEventAction stereotyped with the GaWorkloadEvent represents an event triggering an end-to-end flow. The type of an event (sporadic, periodic, etc.) can be specified within the property called *pattern*. If it is periodic, period can be provided as well. In this case

there are four events for each end-to-end flow. This model refers to the data driven activation. To represent the TD there would be one AcceptEventAction per one functional entity.

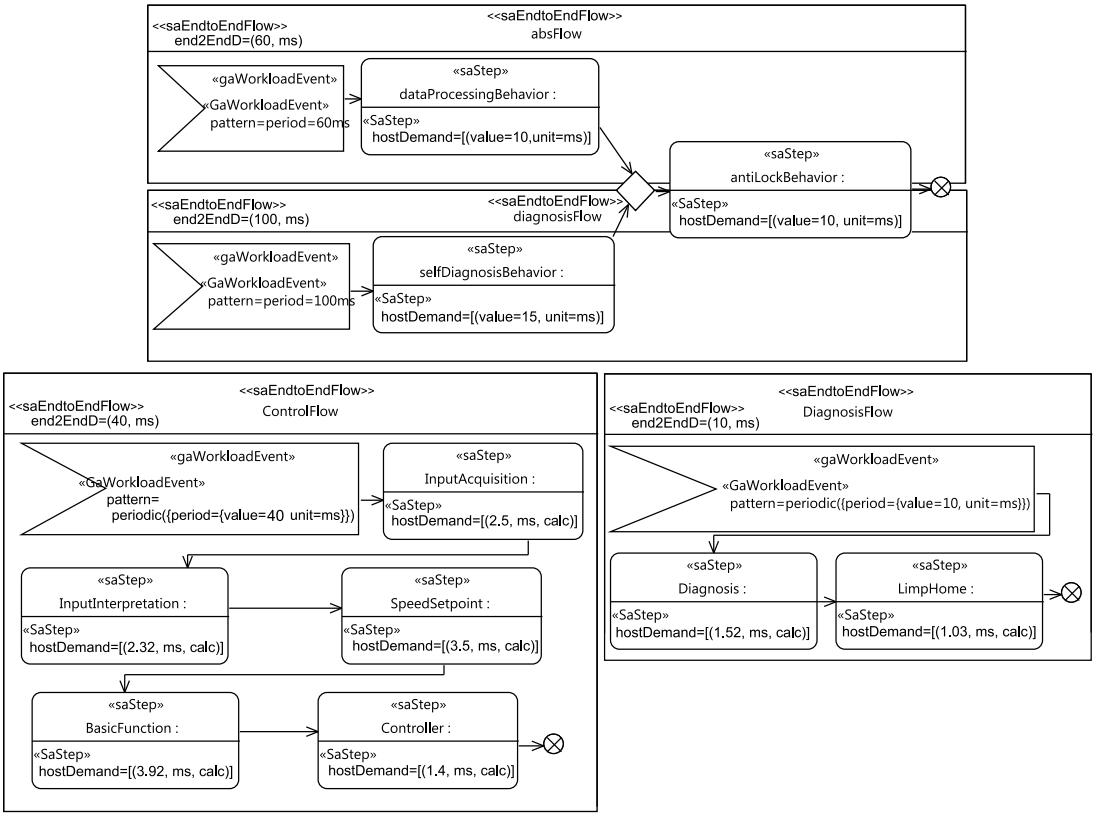

Figure 5.27. Analyzable Model representing System Behavior under the Analysis Viewpoint

In addition to the flows specification the analysis viewpoint contains also the specification of the platform resources, i.e. χ. The types of the hardware elements are modeled using the SysML Block diagram and the Block concept coming from this language. Additionally to differentiate between the types of hardware elements (sensors, actuators, etc.) MARTE is used. The Figure 5.28 presents the model of the hardware types specified under the analysis viewpoint. The hardware topology specification which completes the hardware platform is modeled using the SysML Internal Block diagram. The model of hardware topology (example on the Figure 5.29) presents specific instances of hardware element types and a way in which they communicate.

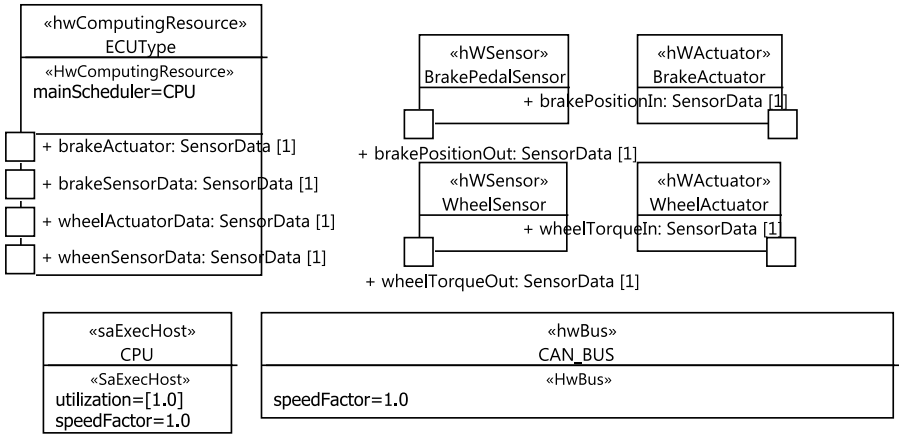

Figure 5.28. Model of Hardware Types specified within the Analysis Viewpoint

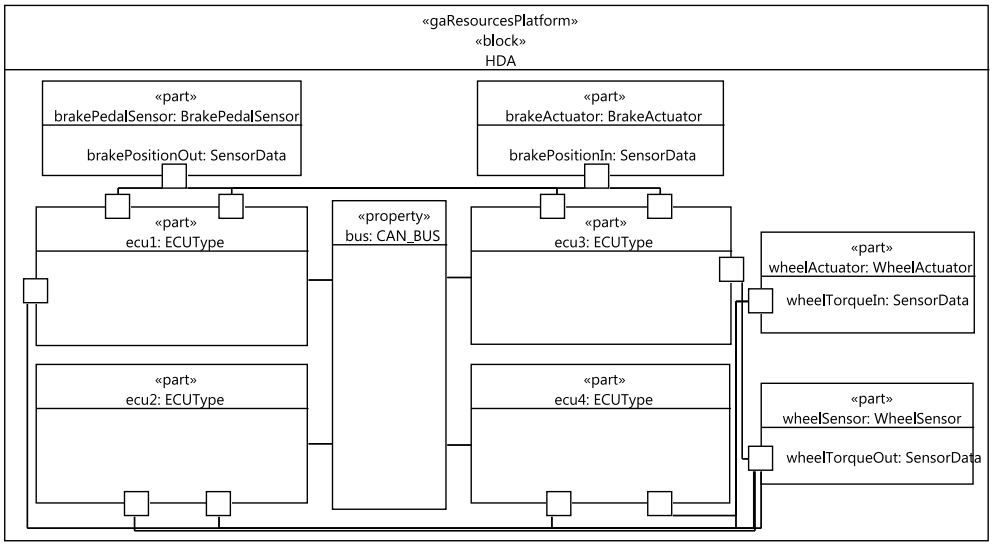

Figure 5.29. Model of Hardware Prototypes specified within the Analysis Viewpoint

The last three figures represent the model of the partial analysis context and hence specify the input for the optimization. In order to complete it the deployment needs to be specified. Therefore the model needs to provide information about the allocation of functional entities/signals to ECUs/BUSes, their partitioning in OS tasks/messages and priorities assignment which refers to the ∂ (see Definition 5.1). This additional information is provided in the activity diagram, i.e. previous workload specification such as this from the Figure 5.27 is enriched with the additional modeling concepts (see Figure 5.30). The functional entities/signals allocation is specified using the property *host* of the stereotypes SaStep and SaCommStep. The OS tasks and messages are represented with the ActivityPartition stereotyped with the SaStep. The partitioning is specified

through the UML as ActivityPartition can contain named elements (property *inPartition*). In this case these will be call behavior actions and control flows relating to functional entities and signals. The SaStep and SaCommStep have property *priority* which holds the priority of the OS task/message. Ordering is defined through the appearance on the list of the named elements contained by the ActivityPartition.

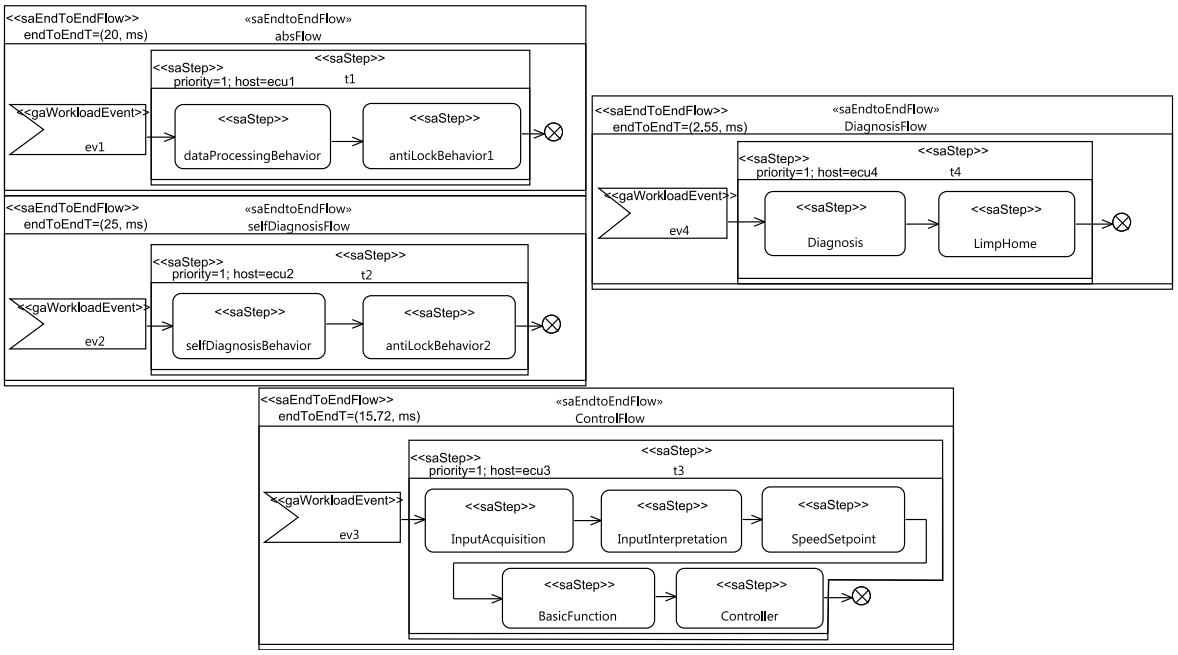

Figure 5.30. Model of a Complete Analysis Context containing specification of the Allocation, Partitioning and Scheduling.

5.3.5. Optimization Viewpoint

Search for an optimized configuration requires provision of additional information such as a definition of exploration dimensions, constraints for an exploration or metrics of interest expressing optimization goals. In the context of the MDE it is a rational choice to include "exploration/optimization information" in a form of a model (called in this work optimization model) that can be then manipulated or interpreted by the optimization tool or built-in optimization engine. It can also help to trace the type of optimizations applied throughout the entire design process. This approach has been used for the specification of concerns related to the analysis such as it was done in the analysis viewpoint (see 5.3.4). There the inclusion of

additional modeling concepts coming in a form of the SysML/MARTE language empowered the handling of analysis concerns throughout a development process by specifying analysis models.

Concerns targeted by the optimization viewpoint refer to the DSE and interest in optimal configuration of system architectures using techniques such as those defined in the chapter 4. The main challenge lies in the modeling of optimization objectives, i.e. provision of an optimization model. As for analysis concerns MARTE delivers necessary concepts to create analysis model, there are no modeling constructs embedded in this language referring to the optimization. This shortage is the main motivation behind the definition of the GOM (Generic Optimization Modeling) profile extending and reusing MARTE. Extension concerns the optimization related concepts whereas reuse refers to the modeling of constraints and non-functional properties which are covered by the MARTE and which enable to establish partial and complete analysis context. The GOM can be then used to build optimization models. Together with the specification of the UML extension for the GOM and the GOM itself, this subsection presents an example of an optimization model on top of which one of the optimization techniques from the chapter 4 can be run to configure the considered use-case.

Specification of the GOM

The central part of the GOM is the optimization context (*OptimizationContext)* defined below (see Definition 5.3). The partial analysis context which is part of this definition was formalized before (see subsection 5.3.4).

Definition 5.3 – Optimization Context: *The optimization context is defined as a quadruple* $\Theta = (\Lambda^{\boldsymbol{I}}, \Xi, \vartheta, \theta)$ *where* $\Lambda^{\boldsymbol{I}}$ *represents partial analysis context,* $\Xi = \{\Xi_1, \Xi_2, \dots, \Xi_n\}$ *exploration parameters,* $\vartheta = \{\vartheta_1, \vartheta_2, \dots, \vartheta_n\}$ *constraints and* $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ *optimization objectives.*

The set of exploration parameters Ξ in other words relates to the decision variables, i.e. elements of the architecture that will be explored. These can be priorities assignment but also the overall deployment, etc. The constraints ϑ come in two forms. The first type of constraints, called exploration constraints restrict the space of possible solutions. An example would be the allocation constraint preserving certain allocation configurations. The second type, are constraints inherited from the analysis context. An example is the latency constraint which needs to be respected by the found architecture configuration. The last but not least are the optimization

objectives θ. There might be multiple objectives driving the search for the best solution, i.e. multi-objective optimization. Single objective θ_i is expressed through the specification of an objective function f_{θ_i}.

The following figures present a metamodel of the GOM and the UML profile specification. The Figure 5.31 contains the core concept, i.e. the *OptimizationContext* which refers to the Θ.

The OptimizationContext holds a reference to the *GaAnslysisContext* coming from the MARTE and referring to the partial analysis context, i.e. Λ^I (property *input*). The partial analysis context is a subject for design space exploration and optimization. The final product of running the DSE is another instance of the GaAnalysisContext related to the input instance of the partial analysis context but with the additional information resulting from the DSE (property *result*). For example input analysis context might be missing information about the allocation, specified by setting the property *host* of the *SaStep*. The resulting analysis context in turn will contain this information.

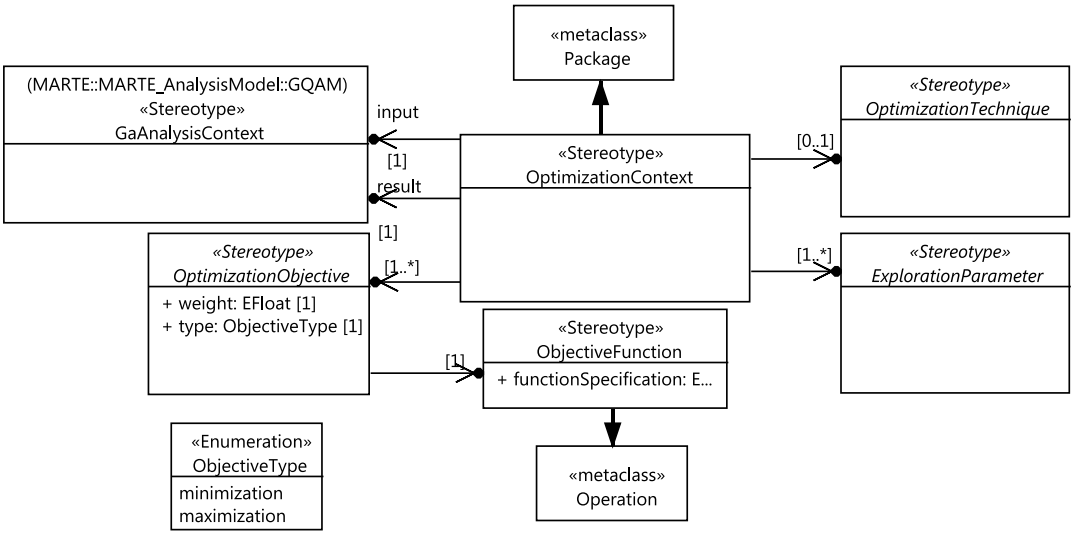

Figure 5.31. UML Profile for the Optimization Context

The next referenced element is the *ExplorationParameter*, i.e. Ξ_i. There needs to be at least one parameter of exploration. This work differentiates few types of exploration parameters (see Figure 5.32). These are the allocation (*Allocation*), specification of a memory protection mechanism (*MemoryProtectionSpecification*), etc. Other types of exploration parameters in

addition to those visible on the Figure 5.32 are possible, presenting a challenge for further extension of the GOM.

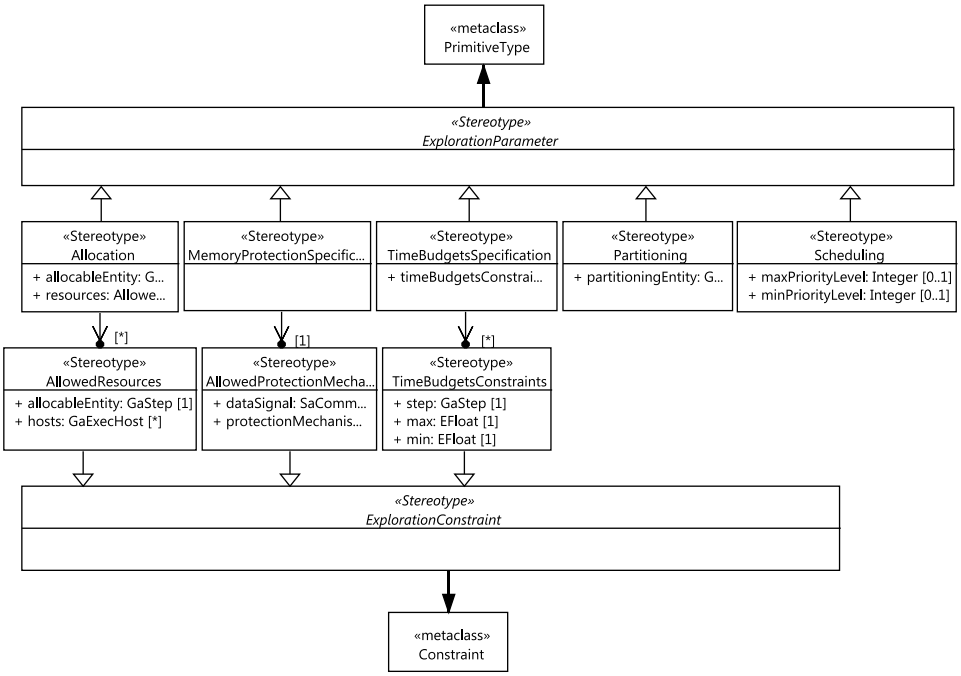

Figure 5.32. UML Profile for the Exploration Parameters

Following is the *OptimizationObjective* which corresponds to the θ_i. As the optimization might be multi-objective, each OptimizationObjective can be assigned a *weight* signifying its importance in regards to other objectives. The Figure 5.33 presents few types of optimization objectives. For instance *E2EResponses* refers to the optimization of end-to-end responses. It contains a property *end2endFlows* which is used to specify the flows of interest, i.e. those end-to-end flows which are the subject for response times optimization. Similarly as in the case of exploration parameters, specification of other objectives types should be considered in the evolution of the GOM. Optimization objective is also characterized by property *type* which defines whether the interest of this objective is to maximize or minimize the objective function.

ObjectiveFunction quantifies how much a specific architecture configuration fulfills the given objective. The property *functionSpecification* serves to provide a mathematical expression of an objective function.

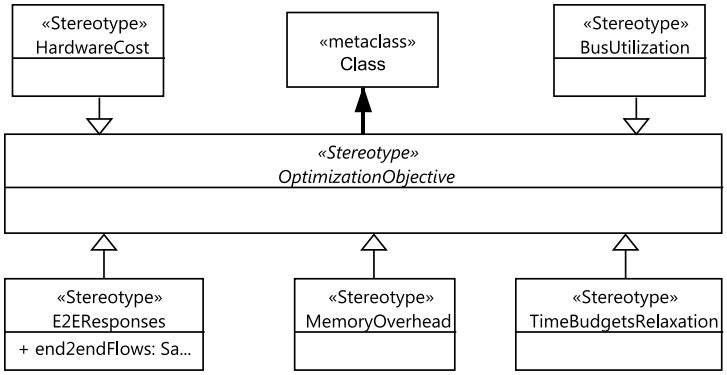

Figure 5.33. UML Profile for the Optimization Objective

The two types of constraints $\mathcal{G}$, i.e. exploration and analysis constraints are represented respectively with the *ExplorationConstraint* stereotype and within the analysis context with the properties of the MARTE stereotypes. Concerning the last an example would be the EndToEndD of the SaEndToEndFlow. This constraint needs to be taken into account during the exploration. There exists a wide range of exploration constraints (see the Figure 5.32) corresponding to the different types of exploration parameters. For instance the constraint called *AllowedResources* is referenced by the exploration parameter *Allocation*. This constraint serves to reduce the number of allocation resources from the all specified within the analysis context to only a specific subset. This limitation is modeled for a particular allocable entity. One of the examples could be a runnable entity which can be allocated only to those ECUs from all available which are connected to a specific sensor.

The last and optional might be specification of an optimization technique that will be used to explore and configure an input architecture. This might prove to be extremely useful in the attempt to generate an input for external optimization tools and hence automate the design process. The abstract stereotype *OptimizationTechnique* should be extended by the definition of a stereotypes relating to the particular optimization techniques. An example of such extension is the stereotype *GeneticAlgorithm*. It provides additional properties specific for the configuration of the GA run. These are the specification of the initial population size (*initialPopulationSize*), definition of the stop condition (*stopCondition*), number of iterations for the GA (*nbOfIterations*) and information about the used crossover and mutation operators (*crossoverOperator*, *mutationOperator*).

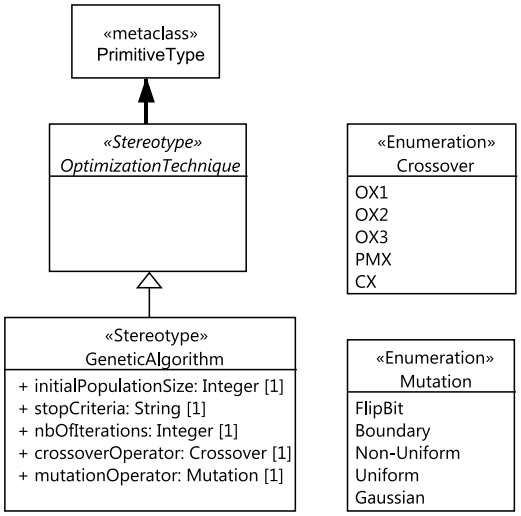

Figure 5.34. UML Profile for the Optimization Technique based on the Genetic Algorithms

The Figure 5.35 represents an optimization model created under the optimization viewpoint. This model is a complete input necessary to run an optimization technique such as it was done in the subsection 4.4.4 on a use-case from the Figure 4.8. The input analysis context (*PartialAnalysisContext*) this is the one from the Figure 5.27. There are three objectives of this optimization which refer to the allocation, partitioning and scheduling, hence in this case the overall deployment as defined for the data driven activation model will take place. The optimization objective concerns the end-to-end responses. The objective function is expressed as a sum of end-to-end flows latencies which is a subject for minimization. The result of optimization is another analysis context here named *AnalysisContext*. The resources platform remains unchanged. What changes is the specification of a workload behavior which additionally to the specification of the end-to-end flows contains now the specification of an allocation and the definition of OS tasks together with their priorities. The Figure 5.30 is a generated, configured and optimized workload behavior.

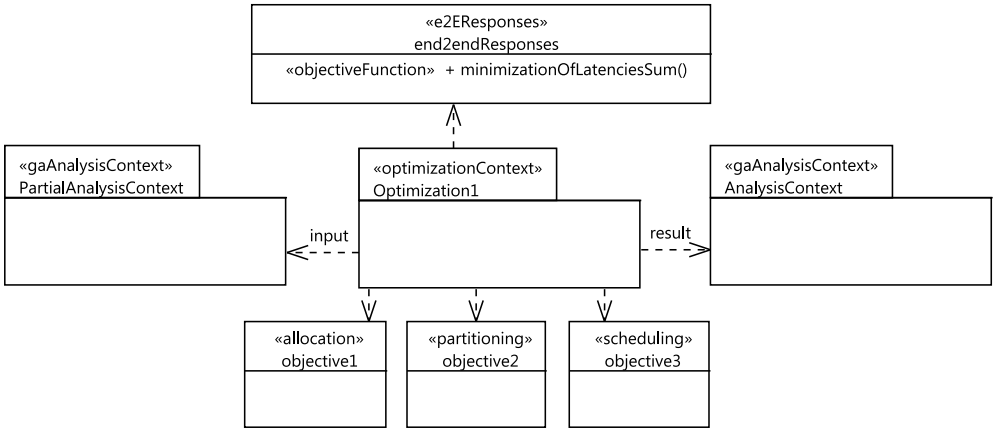

Figure 5.35. Optimization Model created under the Optimization Viewpoint

5.4. Interoperability Viewpoints

Interoperability between the tools is among the major issues in the design of a tools chain where each tool is covering a subset of activities from the all identified within the development process. The definition of the automotive architecture framework should ease the exchange of information between the different tools where each might cover only a subset of viewpoints. The proposed instance of the automotive architecture framework, i.e. the AFfMAO covers all the hitherto specified viewpoints, however we are aware that the AAF should be enriched with other types of viewpoints. These can be safety, simulation or driver perspective related viewpoints. Then an exchange even for the AFfMAO would be unavoidable. Thus the provision of the interoperability viewpoints in the specification of the AAF is desirable. This work accounts for two interoperability viewpoints.

Functional Level Exchange

This viewpoint serves to exchange models developed under the viewpoints of the feature, function and design layer (except the analysis and optimization viewpoint models). These models are expressed with the EAST-ADL2 language. Therefore the language used by the Functional Level Exchange viewpoint is the EAXML (EAST-ADL XML) [100] which is a standardized exchange format for the EAST-ADL2 models.

Technical Level Exchange

The concern of this viewpoint is to exchange models developed under the viewpoints of the technical layer. These models as presented are expressed with the AUTOSAR language.

Consequently the ARXML (AUTOSAR XML) [101] defined by the AUTOSAR consortium is the language used by this viewpoint.

5.5. Correspondence Rules

The correspondence rules are used to enforce relations within an architecture description or between architecture descriptions. This subsection specifies correspondence rules within an architecture description, enabling to transit from the design level models to the technical level models. Following are the correspondence rules used to construct the analysis context out of the EAST-ADL2 model or the AUTOSAR model. This implicitly enables to analyze, explore, configure and optimize the EAST-ADL2 and AUTOSAR models.

5.5.1. EAST-ADL2 and AUTOSAR

Correspondence rules between the design and technical level are significant, as they enable guided generation of the technical level elements. The main idea is to support and speed up the development process by generating the technical level. Naturally it is not possible to generate a complete AUTOSAR model as upper layers abstract away most of the technical level information. However preliminary AUTOSAR model can be produced due to the direct relations of certain EAS|T-ADL2 and AUTOSAR concepts. The Table 5.10 relates the main artifacts of the EAST-ADL2 and the AUTOSAR.

System Aspect	**Design Level – EAST-ADL2**	**Technical Level - AUTOSAR**
Functional/Software Specification	DesignFunctionPrototype typed with a DesignFunctionType with a property *isAtomic* set to false.	SwComponentPrototype
	DesignFunctionType	SwComponentType
	FunctionalDevice	SensorActuatorSwComponentType
	FunctionClientServerPort	Port (RPortPrototype or PPortPrototype) with an interface set to ClientServerInterface. (RPortPrototype represents in this case a client port, PPortPrototype represents a server port).
	FunctionFlowPort	Port with an interface set to SenderReceiverInterface
	FunctionConnector	SwConnector

System Aspect	Design Level – EAST-ADL2	Technical Level - AUTOSAR
	FunctionClientServerInterface	ClientServerInterface
	Operation	ClientServerOperation
	Allocation	SwcToEcuMapping
Behavioor	DesignFunctionPrototype typed with a DesignFunctionType with a property *isAtomic* set to true.	RunnableEntity
Hardware	Node	ECU
	HardwareComponentPrototype typed with an element stereotyped with the Node	ECUInstance
	Sensor	SensorHw
	Actuator	ActuatorHw
	LogicalBus	CanPhysicalChannel/FlexrayPhysicalChannel
	HardwarePort	HwPort
	HardwarePin	HwPin

Table 5.10. Correspondence Rules between the Design and Technical Level Viewpoints

5.5.2. EAST-ADL2 and Analyzable Model

The main reason for defining the correspondence rules between the EAST-ADL2 and the analysis context is to enable the analysis and exploration of the functional models. For instance to evaluate the allocation at the EAST-ADL2 level if given or to find an allocation (such as in the subsection 4.6.1) the functional model should be transformed to the analysis context. Also the opposite way in order to apply the results of exploration on the EAST-ADL2 model, the correspondence rules needs to be identified.

EAST-ADL2	SysML & MARTE
DesignFunctionPrototype	CallBehaviorAction stereotyped with the MARTE SaStep
FunctionConnection	ControlFlow stereotyped with the SaCommStep
EventFunctionFlowPort, EventFunctionClientServerPort	AcceptEventAction stereotyped with the GaWorkloadEvent
PeriodicConstraint and its property *period*	GaWorkloadEvent and its property *period*
ExecutionTimeConstraint	SaStep and its property *hostDemand*

EventChain	ActivityPartition stereotyped with the MARTE SaEndToEndFlow
ReactionConstraint	Property end2EndD of a stereotype SaEndToEndFlow
Node	SysML::Block stereotyped with the MARTE::HwComputingResource
LogicalBus	SysML::Block typed with the MARTE::HwBus
Sensor/Actuator	SysML::Block stereotyped with the MARTE::HwSensor/HwActuator
HardwareComponentPrototype typed with the Node	SysML::Part typed with the SysML::Block stereotyped with the MARTE HwComputingResource
HardwareComponentPrototype typed with the Sensor/Actuator	SysML::Part typed with the SysML::Block stereotyped with the MARTE::HwSensor/HwActuator

Table 5.11. Correspondence Rules between the EAST-ADL2 Model and the Analyzable Context

5.5.3. AUTOSAR and Analyzable Model

The motivation behind the correspondence rules between the AUTOSAR and the analysis context is similar as in the case of the correspondence rules defined in the previous subsection. Namely, they serve to transform the AUTOSAR model to the analysis context and hence enable the analysis and exploration of AUTOSAR models.

AUTOSAR	**SysML & MARTE**
RunnableEntity	CallBehaviorAction stereotyped with the MARTE SaStep
TDEventSwcInternalBehavior	AcceptEventAction stereotyped with the GaWorkloadEvent
PeriodicEventTriggering and its property *period*	GaWorkloadEvent and its property *period*
TimingDescriptionEventChain	ActivityPartition stereotyped with the MARTE SaEndToEndFlow. Based on the specification of the events chain the control flow is constructed, i.e. instances of the ControlFlow stereotyped with the SaCommStep are created.
LatencyTimingConstraint	Property end2EndD of a stereotype SaEndToEndFlow
ECU	SysML::Block stereotyped with the MARTE::HwComputingResource
Cluster and PhysicalChannel	SysML::Block typed with the MARTE::HwBus

SensorHw/ActuatorHw	SysML::Block stereotyped with the MARTE::HwSensor/HwActuator
ECUInstance	SysML::Part typed with the SysML::Block stereotyped with the MARTE HwComputingResource
Property typed with the SensorHw/ActuatorHw	SysML::Part typed with the SysML::Block stereotyped with the MARTE::HwSensor/HwActuator

Table 5.12. Correspondence Rules between the AUTOSAR Model and the Analyzable Context

5.6. Conclusions

This chapter presented the automotive architecture framework and its specific instance called in this work AFfMAO. The specification of the AAF comprise the definition of the viewpoints and its structuring into the abstraction layers. Each viewpoint is defined through its model kinds which accommodate specification of the modeling languages and modeling techniques. Notable for this framework is the tackling of concerns related to the analysis and optimization by defining the analysis and optimization viewpoints. The analysis viewpoint incorporates the SysML and MARTE languages as enablers for the analysis such as the schedulability analysis test. The optimization viewpoint as its notation uses the optimization profile also defined in this chapter. This and the correspondence rules defined between the EAST-ADL2 and the AUTOSAR models permit to integrate the analysis and optimization in the design flow which corresponds to the concern of leading a qualitative design. The integration was also eased through the use of the UML profiles mechanism. This facilitates the development of the transformations based on the predefined correspondence rules but also the integration of new modeling concepts as either structural or behavioral models can be simply annotated with additional information central for running the analysis or optimization. Lastly, this chapter described correspondence rules between the design and technical layers to support an idea of a seamless development flow.

In general the AAF as presented here responds to the crucial concern, i.e. qualitative design and modeling of the architecture at the system level. It is however clear that to cover all the aspects of the design and other possible concerns, the current definition of the AAF should be extended with additional viewpoints. For instance simulation or safety are the substantial concerns which haven’t been addressed by this work.

6. Conclusion

6.1. Summary

This thesis proposed a framework for the modeling analysis and optimization of automotive architectures. It builds upon established EAST-ADL2 and AUTOSAR methodology and concept of Architecture Framework. Its main goal is to support design activities in order to produce system architecture of high quality. This is obtained first through the use of models and their composition in a set of viewpoints referring to different design concerns. This makes the overall design more comprehensible as models abstract from the low level implementation details. Secondly it is the application of analysis techniques to assess the feasibility of the architecture in regards to the predefined constraints. These concern the timing constraints, i.e. preservation of end-to-end deadlines for transactions or paths. Thirdly the quality of design is improved by engaging the optimization techniques. Due to the revealed shortcomings in current support for the automated design space exploration, this thesis proposed techniques for the deployment and time budgeting paying particular attention to the problem of scalability. This provoked an idea for using powerful evolutionary algorithms further enhanced with an adoption of two strategies: divide and conquer and iterative improvement.

Chapter 2 presented standards such as Architecture Framework, AUTOSAR or CAN protocol and the methodology for designing automotive architectures based on the EAST-ADL2 and the AUTOSAR. Its intent was to provide detailed picture of the context and the fundamentals necessary to comprehend the main, tackled challenges and developed contributions.

Chapter 3 listed the challenges identified for the automotive domain paying particular attention to the problem of configuration of automotive architectures and architecture description specification.

Chapter 4 refers to the optimization capabilities of the advertised framework. It demonstrated a set of techniques for design space exploration. First it formalized two models of computation, i.e. data driven and time driven and for each of them problem of deployment. In the case of data driven the deployment consists of specification of allocation of runnables, their partitioning in OS tasks and scheduling, i.e. assignment of priorities to tasks. The deployment for time driven systems expands with two additional sub-problems which is the ordering of runnables inside the OS tasks (as in this case the runnables of different paths are allowed to reside on the

same task) and choice of a protocol for the protection of shared resources. Through the study over the related work it was shown that for such defined deployment the current techniques don't treat all the deployment dimensions holistically. Consequently this work contributed by presenting holistic approaches for solving the deployment, based on the genetic algorithms. Approach for data driven activation model uses end-to-end deadlines as the exploration timing constraint. For the optimization objective it refers to two timing performance metrics, the sum of response times and the minimal slack. Similarly, the timing plays the role of a constraint and optimization objective when optimizing deployment of architectures based on time driven assumptions. However as the specification of protection mechanism impacts the used memory overhead as well as timing, the second optimization objective, referring to the memory was defined.

Also in this chapter large effort was dedicated to the scalability issue. Due to the NP hard nature of the deployment problem the runtime for the holistic approaches, although based on the powerful genetic algorithms increases exponentially. This prevents the qualitative optimization of large sized input architectures. Consequently this work applied two additional heuristic strategies i.e. the iterative improvement and divide and conquer.

Finally this chapter exhibited a new idea for the specification of time budgets. It is built upon three distinct assumptions that the time budgets represent constraint on the runnables WCET, the input for this is architecture not yet synthetized and the objective is to relax the time budgets finding at the same time the deployment which respects the timing constraints. As the deployment is an inherent part of the time budgeting process, this work reuses the previous contributions and combines them with the algorithm for time budgets assignment.

Chapter 5 elaborated on the modeling capabilities of the advertised framework and how the three activities, i.e. modeling, analysis and optimization can be integrated in the model based design of automotive systems. It presented an instance of the Architecture Framework for the automotive (Automotive Architecture Framework). Its specification comprised the definition of the viewpoints and its structuring into the abstraction layers. Each viewpoint was defined through its model kinds which accommodate specification of the modeling languages and modeling techniques. The main languages used to define the system architecture these are the EAST-ADL2 and the AUTOSAR.

The analysis and optimization was integrated with the modeling through the definition of analysis and optimization viewpoints. The first one incorporates the SysML and MARTE languages. To express the concerns of the optimization viewpoint, this work defined the UML profile. This and the correspondence rules defined between the viewpoints using notation of the EAST-ADL2/AUTOSAR, and analysis and optimization viewpoints permit to integrate the analysis and optimization in the design flow which corresponds to the concern of leading a qualitative design. The integration was also achieved through the use of the UML profile mechanism. This facilitates the development of the transformations based on the predefined correspondence rules but also the integration of new modeling concepts as either structural or behavioral models can be simply annotated with additional information central for running the analysis or optimization

6.2. Future Work

There exist many possible scenarios for further extension of this work. First and natural extension would be an inclusion of additional concerns in the definition of the AAF such as those related to the non-functional characteristics of an architecture description. Please note that this work paid particular attention to the timing and memory. It is however substantial to refer also to safety, cost, power consumption, etc. Their consideration is not straightforward due to significant cross-dependencies between all of them. Work on these aspects is advanced but it might occur that certain shortcomings exist, similarly as it was for the deployment and consideration of timing and memory. Nevertheless the AAF would have to be extended with additional viewpoints for which definition of supplementary modeling constructs would be indispensable. In this respect, example of a useful related work is [102] or [103]. The first one provides a viewpoint to account for safety through the specification of a replication strategy for safety critical components. Authors define UML profile which captures concepts of *replica*, *replication strategy* and others. The second work extends MARTE to grasp the information necessary for reasoning about the power consumption. Definitely further evolution of the AAF should be stimulated by the cooperation among the automotive partners to grasp their common concerns. This work might encourage the reflection about the advantages of having common AAF definition and hence trigger some discussions on that issue.

Qualitative consideration of new properties (i.e. safety, cost, etc.) would require to expand with proposed design space exploration techniques. In consequence this would necessitate further

evolvement of the optimization profile. This means adding new exploration parameters and constraints, optimization objectives and hence new metrics, and most probably optimization techniques with their parameters.

Concerning the proposed techniques for deployment, it is believed that additional gain can be achieved from further improvements. It would be interesting to evaluate other choices of initial population or evolution operators for the genetic algorithms. Although the runtimes of presented algorithms are already compatible with problems of industrial size we could also try to profit from the powerful grid machines. This requires parallel implementation of the genetic algorithms so they can be run on distributed computing architectures.

Finally the resulting AFfMAO framework can be restructured to embed other domains, not just the automotive. For instance avionic or train systems share lots of common concerns with the automotive systems. These are also highly critical systems therefore analysis of their behavior, especially timing behavior is of a high importance. Consequently the Qompass framework can be successfully reused within these two domains. This is especially because it builds upon general purpose modeling language, i.e. SysML which is applicable to the wide range of systems due to its generic nature. Also the MARTE profile has no tight connections to any particular domain, except that it targets group of real-time systems to which avionic or train systems belong. In fact, the desire to apply Qompass to other domains was the main factor that determined the choice of the SysML and MARTE languages.

Automotive domain is evolving rapidly if we consider the emergent of new solutions but also new challenges. This makes it an interesting and practical background for further scientific advancements. I believe that this framework is a good base reference to continue with research activities related to modeling, analysis and optimization of automotive systems.

References

[1] ISO/IEC/(IEEE), "ISO/IEC/IEEE 42010:2011 : Systems and software engineering - Architecture description," 11 2011.

[2] T. Dittel and H.-J. Aryus, "How to "survive" a safety case according to iso 26262," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, E. Schoitsch, Ed. Springer Berlin Heidelberg, 2010, vol. 6351, pp. 97–111.

[3] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/-1134285.1134292

[4] M. Broy, I. Krüger, A. Pretschner, and C. Salzmann, "Engineering Automotive Software," *Proceedings of the IEEE*, vol. 95, no. 2, 2007.

[5] AUTOSAR, http://www.autosar.org/.

[6] *EAST-ADL Domain Model Specification V2.1.11*, 05 2013.

[7] D. Ku and G. Micheli, "EnglishDesign space exploration," in *EnglishHigh Level Synthesis of ASICs under Timing and Synchronization Constraints*, ser. The Springer International Series in Engineering and Computer Science. Springer US, 1992, vol. 177, pp. 83–111. [Online]. Available: http://dx.doi.org/10.1007/978-1-4757-2117-1_5

[8] *AUTOSAR Specification of Timing Extensions*, AUTOSAR Std. 1.2.0. [Online]. Available: http://www.autosar.org/download/R4.0/-AUTOSAR_TPS_TimingExtensions.pdf

[9] S. Anssi, "Methodology for model-based timing analysis process for automotive systems," Ph.D. dissertation, l'Université Paris-Sud, 2011.

[10] *Unified Modeling Language Superstructure v2.3*, OMG Std.

[11] SysML, http://www.sysml.org/.

[12] *UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, Version 1.0, formal/2009-11-02*, OMG, November 2009. [Online]. Available: http://-www.omgmarte.org/

[13] OSEK/VDX, http://www.osek-vdx.org/.

[14] *Specification of Operating System V5.2.0*, 10 2013.

[15] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[16] ——, "Event-triggered versus time-triggered real-time systems," in *Operating Systems of the 90s and Beyond*, ser. Lecture Notes in Computer Science, A. Karshmer and J. Nehmer, Eds. Springer Berlin Heidelberg, 1991, vol. 563, pp. 86–101. [Online]. Available: http://dx.doi.org/10.1007/BFb0024530

[17] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[18] H. Kopetz and G. Grunsteidl, "Ttp-a protocol for fault-tolerant real-time systems," *Computer*, vol. 27, no. 1, pp. 14–23, 1994.

[19] O. Scheickl, "Timing constraints in distributed development of automotive real-time systems," Ph.D. dissertation, Technische Universität München für Informatik, 2011.

[20] U.S. Government Department of Defense, "The DoDAF Architecture Framework Version 2.02." 2009.

[21] MODAF partners, "MOD Architectural Framework Technical Handbook, Version 1.0," August 2005.

[22] EAST-EEA, http://www.itea2.org/public/project_leaflets/EAST-EEA_results_oct-04.pdf/.

[23] ATESST, http://www.atesst.org/.

[24] D. Chen, H. Lönn, F. Törner, and H. Blom, "Advancing traffic efficiency and safety through software technology (atesst) deliverable d2.2.2," Tech. Rep., January 2008. [Online]. Available: http:/www.atesst.org/

[25] M. Weber and H. Lönn, "Methodology guideline when using east-adl2," Tech. Rep. Deliverable D5.1.1, June 2010. [Online]. Available: http://www.atesst.org/home/liblocal/-docs/ATESST2_Deliverable_D5.1.1_V1.1.pdf

[26] *AUTOSAR Methodology*, AUTOSAR Std. [Online]. Available: http://www.autosar.org/-download/AUTOSARMethodology.pdf

[27] T. N. Qureshi, D.-J. Chen, H. Lönn, and M. Törngren, "From east-adl to autosar software architecture: A mapping scheme," in *ECSA*, 2011, pp. 328–335.

[28] *International Standards Organization, ISO/DIS 26262:2009 - Draft International Standard Road Vehicles - Functional Safety"*, http://www.iso.org, Std.

[29] *TIMMO-2-USE Timing Model - Tools, algorithms, languages, methodology, USE cases*, TIMMO-2-USE Partners, October 2012. [Online]. Available: http://www.timmo-2-use.org/

[30] *dSpace*, http://www.dspaceinc.com/.

[31] Vector, http://www.vector.com/.

[32] S. (SymtaVision), http://www.symtavision.com/symtas.html.

[33] SynDEx, http://www.syndex.org/.

[34] K. Tindell and J. Clark, "Holistic schedulability for distributed hard real-time systems," *Microprocessing & Microprogramming*, vol. 40, pp. 117–134, 1994.

[35] J. C. Palencia and M. G. Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proceedings of the 19th IEEE Real-Time Systems Symposium*, 1998, p. 26.

[36] Q. Zhu, Y. Yang, E. Scholte, M. Di Natale, and A. Sangiovanni-Vincentelli, "Optimizing extensibility in hard real-time distributed systems," in *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2009, pp. 275–284.

[37] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *Computers, IEEE Transactions on*, vol. 39, no. 9, pp. 1175–1185, 1990.

[38] *The Mathworks Simulink and StateFlow User's Manuals*. [Online]. Available: http://-www.mathworks.com

[39] H. Zeng and M. D. Natale, "Efficient implementation of autosar components with minimal memory usage," in *SIES*, 2012, pp. 130–137.

[40] B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Transaction on Software Engineering*, 2012.

[41] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*, April 2011, pp. 71–80.

[42] P. Pop, P. Eles, Z. Peng, and T. Pop, "Analysis and optimization of distributed real-time embedded systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 3, pp. 593–625, 2006.

[43] T. Pop, P. Eles, and Z. Peng, "Design optimization of mixed time/event-triggered distributed embedded systems," in *Proc. of the First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, New York, NY, USA, 2003.

[44] X. He, Z. Gu, and Y. Zhu, "Task allocation and optimization of distributed embedded systems with simulated annealing and geometric programming," *The Computer Journal*, vol. 53, no. 7, pp. 1071–1091, 2010.

[45] I. Bate and P. Emberson, "Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems," in *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006, pp. 221–230.

[46] A. Hamann, R. Racu, and R. Ernst, "Multi-dimensional robustness optimization in heterogeneous distributed embedded systems," in *Proceedings of the 13th IEEE Real Time and Embedded Technology and Applications Symposium*, April 2007.

[47] E. Azketa, J. Uribe, J. Gutierrez, M. Marcos, and L. Almeida, "Permutational genetic algorithm for the optimized assignment of priorities to tasks and messages in distributed real-time systems," in *Proceedings of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2011, pp. 958–965.

[48] S. Kugele, W. Haberl, M. Tautschnig, and M. Wechs, "Optimizing automatic deployment using non-functional requirement annotations," *Leveraging Applications of Formal Methods, Verification and Validation*, pp. 400–414, 2009.

[49] M. Richard, P. Richard, and F. Cottet, "Allocating and scheduling tasks in multiple fieldbus real-time systems," in *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1. IEEE, 2003, pp. 137–144.

[50] C. Bartolini, G. Lipari, and M. Di Natale, "From functional blocks to the synthesis of the architectural model in embedded real-time applications," in *Proc. 11th IEEE Real Time and Embedded Technology and Applications Symposium*, 2005, pp. 458–467.

[51] M. Saksena, P. Karvelas, and Y. Wang, "Automatic synthesis of multi-tasking implementations from real-time object-oriented models," in *Proceedings of 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2000, pp. 360–367.

[52] S. Kodase, S. Wang, and K. Shin, "Transforming structural model to runtime model of embedded software with real-time constraints," in *Proceedings of the conference on Design, Automation and Test in Europe*, 2003, pp. 170–175.

[53] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. Sangiovanni-Vincentelli, "Optimization of task allocation and priority assignment in hard real-time distributed systems," *ACM Transactions on Embedded Computing Systems*, vol. 11, no. 4, pp. 85:1–85:30, 2012.

[54] H. Zeng, M. D. Natale, and Q. Zhu, "Optimizing stack memory requirements for real-time embedded applications," in *ETFA*, 2012, pp. 1–8.

[55] A. Ferrari, M. D. Natale, G. Gentile, G. Reggiani, and P. Gai, "Time and memory tradeoffs in the implementation of autosar components," in *DATE*, 2009, pp. 864–869.

[56] M. Zhang and Z. Gu, "Optimization issues in mapping autosar components to distributed multithreaded implementations," in *International Symposium on Rapid System Prototyping*, 2011, pp. 23–29.

[57] L. Davis, Ed., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[58] B. L. Miller, B. L. Miller, D. E. Goldberg, and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, pp. 193–212, 1995.

[59] A. Mehiaoui, E. Wozniak, S. T. Piergiovanni, C. Mraidha, M. D. Natale, H. Zeng, J.-P. Babau, L. Lemarchand, and S. Gérard, "A two-step optimization technique for functions placement, partitioning, and priority assignment in distributed systems," in *SIGPLAN/SIGBED Conference on Languages, Compilers and Tools for Embedded Systems LCTES*, 2013, pp. 121–132.

[60] S. Anssi, S. Tucci-Piergiovanni, S. Kuntz, S. Gerard, and F. Terrier, "Enabling scheduling analysis for autosar systems," *Object-Oriented Real-Time Distributed Computing, IEEE International Symposium on*, vol. 0, pp. 152–159, 2011.

[61] C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard, "Optimum: a marte-based methodology for schedulability analysis at early design stages," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 1, pp. 1–8, 2011.

[62] E. Wozniak, S. Tucci-Piergiovanni, C. Mraidha, and S. Gerard, "An integrated approach for modeling, analysis and optimization of systems whose design follows the east-

adl2/autosar methodology," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 6(1), no. 276-286, 2013.

[63] A. Mehiaoui, S. T. Piergiovanni, J.-P. Babau, and L. Lemarchand, "Optimizing the deployment of distributed real-time embedded applications," in *RTCSA*, 2012, pp. 400–403.

[64] J. Rox, K. Schmidt, A. Winter, T. Spengler, and R. Ernst, "Estimating and mitigating design risk in a flexible distributed design process," *IEEE Embedded Systems Letters*, vol. 2, no. 2, pp. 35–38, 2010.

[65] *ALL TIMES: D2.23.2 Final prototype of integrated system-level verification methodology*, ALL TIMES Partners, August 2010. [Online]. Available: http://www.mrtc.mdh.se/-projects/all-times/

[66] M. Di Natale and J. A. Stankovic, "Dynamic end-to-end guarantees in distributed real time systems," in *Proceedings of the IEEE Real-Time Systems Symposium*, 1994, pp. 216–227.

[67] R. Gerber, S. Hong, and M. Saksena, "Guaranteeing real-time requirements with resource-based calibration of periodic processe," in *IEEE Transactions on Software Engineering*, vol. 21, July 1995, pp. 579–592.

[68] N. Serreli and E. Bini, "Deadline assignment for component-based analysis of real-time transactions," in *2nd Workshop on Compositional Real-Time Systems, Washington, DC, USA*, 2009.

[69] S. Hong, T. Chantem, and X. S. Hu, "Meeting end-to-end deadlines through distributed local deadline assignment," in *Proceedings of the IEEE Real-Time Systems Symposium*, 2011.

[70] P. Jayachandran and T. Abdelzaher, "Delay composition in preemptive and non-preemptive real-time pipelines," in *Real-Time Systems Journal*, vol. 40, 2008, pp. 290–320.

[71] N. Feiertag, K. Richter, and C. Ficek, "On the decomposition of end-to-end timing requirements in distributed partitioned automotive functions," in *SAE World Congress, Detroit, USA*, 2012.

[72] M. Di Natale and A. L. Sangiovanni-Vincentelli, “Moving from federated to integrated architectures in automotive: The role of standards, methods and tools,” *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.

[73] M. G. Dixit, S. Ramesh, and P. Dasgupta, “Time-budgeting: a component based development methodology for real-time embedded systems,” 2013.

[74] M. G. Dixit, P. Dasgupta, and S. Ramesh, “Taming the component timing: A cbd methodology for real-time embedded systems,” in *DATE*, 2010, pp. 1649–1652.

[75] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, “System level performance analysis - the symta/s approach,” *Computers and Digital Techniques, IEE Proceedings* -, vol. 152, no. 2, pp. 148–166, Mar 2005.

[76] R. Racu, A. Hamann, and R. Ernst, “A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems,” in *18th Euromicro Conference on Real-Time Systems*, 2006.

[77] Simulink, http://www.mathworks.com/products/simulink/.

[78] R. Isermann, J. Schaffnit, and S. Sinsel, “Hardware-in-the-loop simulation for the design and testing of engine-control systems,” *Control Engineering Practice*, vol. 7, no. 5, pp. 643 – 653, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/-S0967066198002056

[79] W.-H. Kwon and S.-G. Choi, “Real-time distributed software-in-the-loop simulation for distributed control systems,” in *Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on*, 1999, pp. 115–119.

[80] M. Graphics, http://www.mentor.com/.

[81] SystemDesk, http://www.dspaceinc.com/en/inc/home/products/sw/-system_architecture_software/systemdesk.cfm/.

[82] TargetLink, http://www.dspace.com/en/pub/home/products/sw/pcgs/targetli.cfm.

[83] D. Developer, http://www.vector.com/vi_davinci_developer_en.html/.

[84] V. toolset, http://www.mentor.com/products/vnd/autosar-products/.

[85] BridgePoint, http://www.mentor.com/products/sm/model_development/bridgepoint/.

[86] Artop, https://www.artop.org/.

[87] Metacase, http://www.metacase.com/.

[88] Systemite, http://www.systemite.se/.

[89] PREEVision, http://vector.com/vi_preevision_en.html.

[90] A. Bauer, M. Broy, J. Romberg, B. Schätz, P. Braun, U. Freund, N. Mata, R. Sandner, P. Mai, and D. Ziegenbein, "Das AutoMoDe-Projekt: Modellbasierte Entwicklung softwareintensiver Systeme im Automobil," *Computer Science – Research and Development*, vol. 22, no. 1, pp. 45–57, 2007.

[91] F. Hölzl and M. Feilkas, "Autofocus 3: a scientific tool prototype for model-based development of component-based, reactive, distributed systems," in *Proceedings of the 2007 International Dagstuhl conference on Model-based engineering of embedded real-time systems*, ser. MBEERTS'07. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 317–322.

[92] S. Voss and B. Schätz, "Deployment and scheduling synthesis for mixed-critical shared-memory applications," in *ECBS*, 2013, pp. 100–109.

[93] M. Broy, M. Gleirscher, S. Merenda, D. Wild, P. Kluge, and W. Krenzer, "Toward a holistic and standardized automotive architecture description," *Computer*, vol. 42, pp. 98–101, 2009.

[94] *Papyrus MDT project webpage: http://www.eclipse.org/modeling/mdt/papyrus/*. [Online]. Available: http://www.eclipse.org/modeling/mdt/papyrus/

[95] Eclipse, http://www.eclipse.org/.

[96] *EAST-ADL UML Profile Specification V2.1.10*, 06 2012.

[97] *Specification of ECU Configuration*, AUTOSAR Std. V3.1.0. [Online]. Available: http://-www.autosar.org/download/R4.0/AUTOSAR_TPS_TimingExtensions.pdf

[98] A. Albinet, L. Queran, B. Sanchez, and Y. Tanguy, "Requirement management fro msystem modeling to autosar sw components," in *Embedded Real Time Software and Systems - ERTS*, 2010.

[99] H. D. E. Ortiz, "An integrated model-driven framework for specifying and analyzing non-functional properties of real time systems," Ph.D. dissertation, l'Université d'Evry, 2007.

[100] *EAST-ADL XML Schema*, MAENAD Std. 3.1. [Online]. Available: http://-www.maenad.eu/public/Deliverables/MAENAD_Deliverable_D4.3.1_V3.1.pdf

[101] *Application Interfaces User Guide*, AUTOSAR Std. V1.3.0. [Online]. Available: http://-www.autosar.org/download/R4.1/AUTOSAR_EXP_AIUserGuide.pdf

[102] S. Tucci-Piergiovanni, C. Mraidha, E. Wozniak, A. Lanusse, and S. Gerard, "A uml model-based approach for replication assessment of autosar safety-critical applications," *IEEE TrustCom/IEEE ICESS/FCST, International Joint Conference of*, vol. 0, pp. 1176–1187, 2011.

[103] M. Hagner, A. Aniculaesei, and U. Goltz, "Uml-based analysis of power consumption for real-time embedded systems," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, Nov 2011, pp. 1196–1201.

Appendix

A. Tool Prototype

This part provides information about the implementation of the AFfMAO. This concerns the platform on top of which it was developed, creation of UML profiles, implementation of transformations and algorithms for analysis and optimization. It also provides links to the videos presenting some of the features of the AFfMAO.

Platform

All the functionality of the AFfMAO was built on top of the Papyrus MDT framework. Papyrus is aiming at providing an integrated environment for editing EMF (Eclipse Modeling Framework) models, in particular supporting UML and related modeling languages such as SysML and MARTE. Papyrus is also convenient to develop custom plugins. This includes definition of UML profiles for Domain Specific Languages (DSL), customization of a palette to display the DSL concepts and customization of Property View, etc. The Figure A.1 shows the layers of the Papyrus MDT.

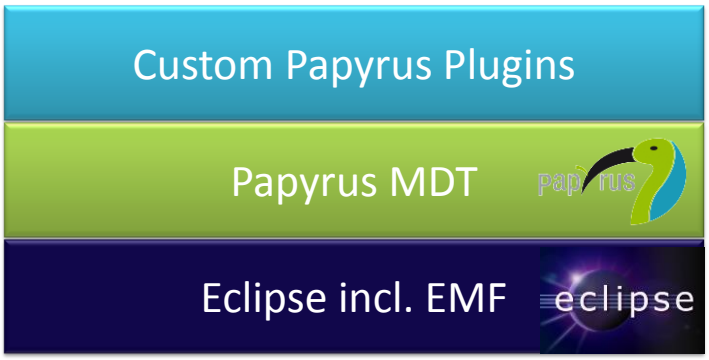

Figure A.1. Layers of Papyrus MDT

New AFfMAO Project

The new AFfMAO project can be created by importing a project template. It contains initial structure of a project, i.e. views corresponding to each of the viewpoints (see Figure A.2). They are also grouped according to the layers as described in section 5.3.

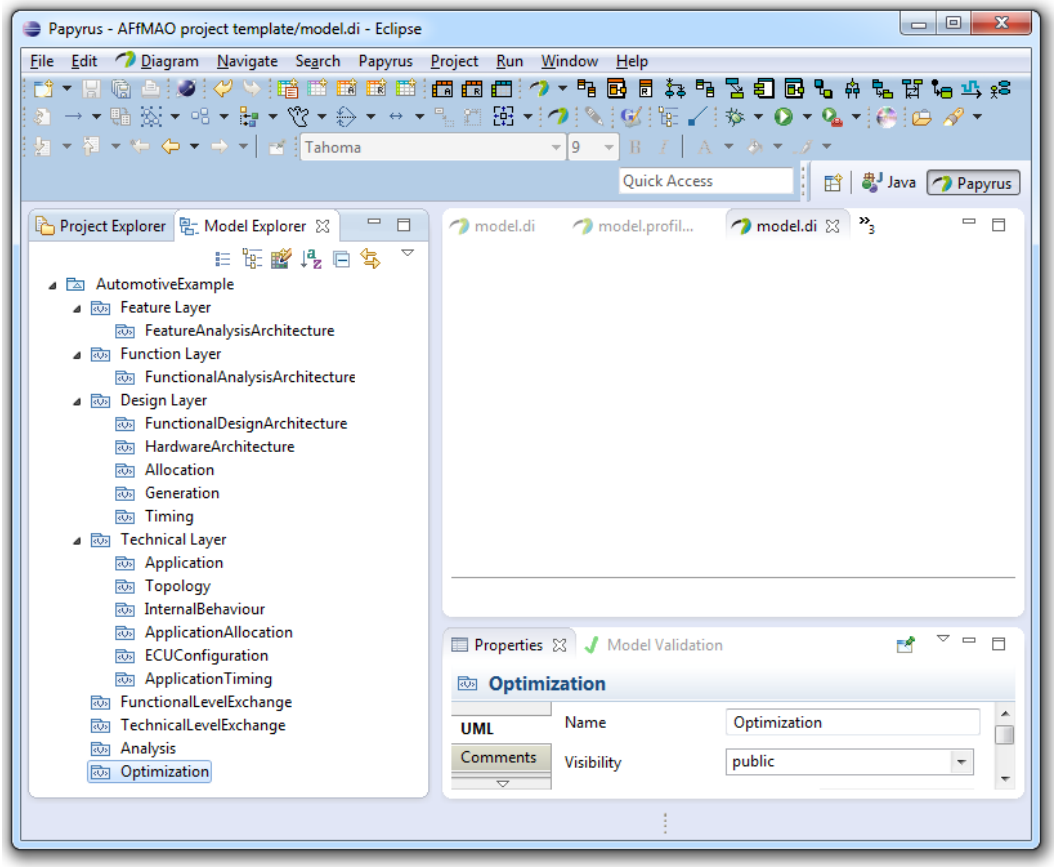

Figure A.2. Generation of AFfMAO Project Template

Creation of UML Profiles

The Papyrus implements mechanism for creating UML profiles. These profiles can be then used by simply importing them to the modeling project. The Figure A.3 shows an example in which EAST-ADL2 profile is being imported.

The AFfMAO is using UML profiles for the EAST-ADL2, MARTE, AUTOSAR, Optimization Metamodel and Generation Metamodel. The UML profiles for the first two languages were already implemented in the Papyrus tool for the needs of the previous projects. For the rest, the UML profiles were implemented explicitly for the needs of the AFfMAO. The Figure A.4 shows part of the UML profile specification done within the Papyrus for the AUTOSAR.

In order to facilitate the modeling, another possible improvement is the customization of a palette. The Figure A.5 shows a palette for the EAST-ADL2.

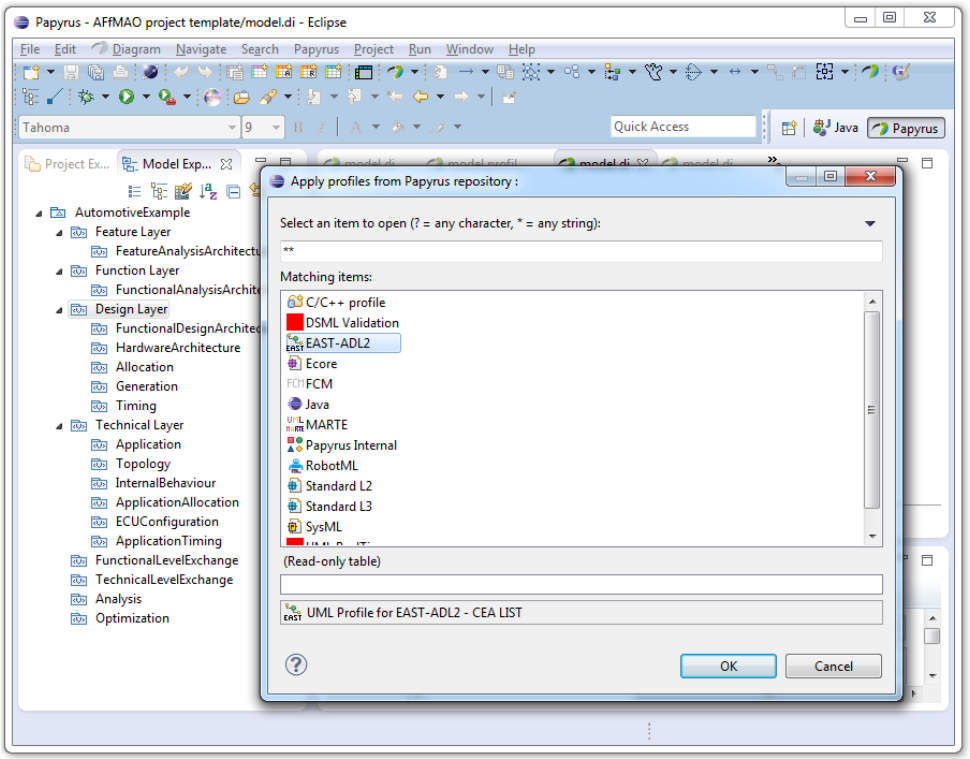

Figure A.3. Applying UML profile for the EAST-ADL2

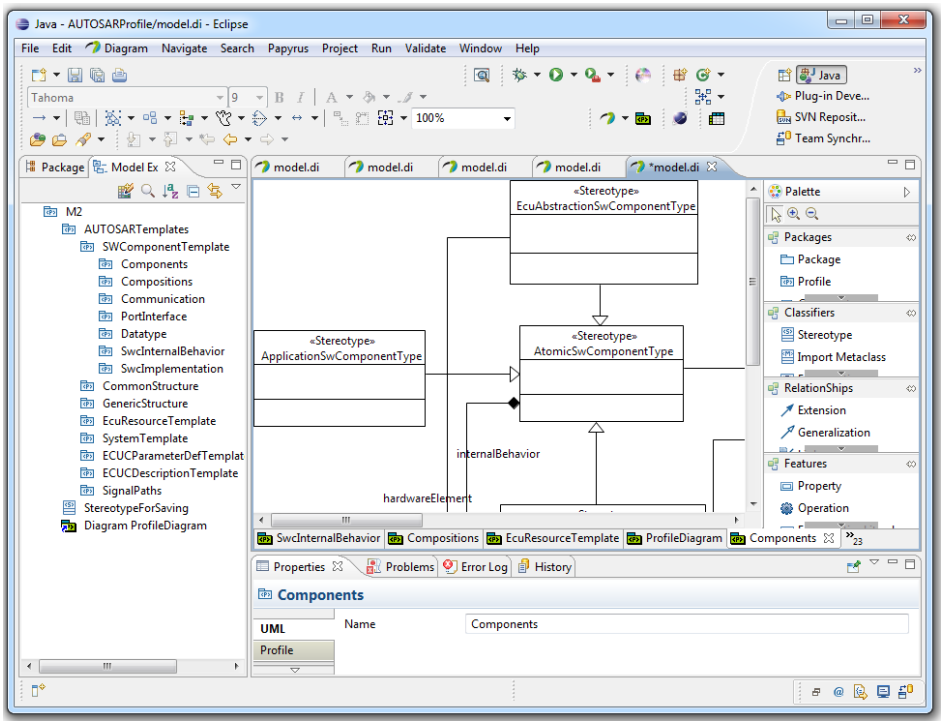

Figure A.4. UML Profile for the AUTOSAR

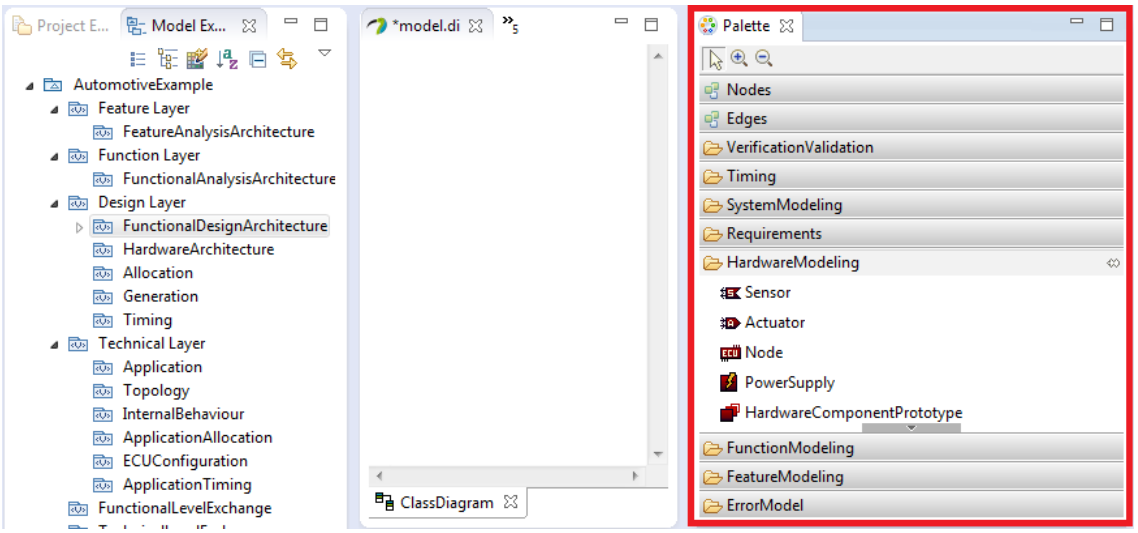

Figure A.5. Palette for the EAST-ADL2

Transformations

The proposed framework integrates few transformation engines called: EAXmlGen, EAQompGateway, ARQompGateway, ARXmlGen, ARGateway. They were all implemented as an Eclipse plugins.

- EAXmlGen – generates an eaxml file from the EAST-ADL2 model created in Papyrus with its UML profile. Its purpose is to serialize and deserialize the EAST-ADL2 model within an eaxml file. The last is an xml file whose content conform to the xml schema of the corresponding EAST-ADL2 release. The generation of eaxml is done in Java. For this purpose it is using the Java API delivered with the EATOP (EAST-ADL Tool Platform - https://projects.eclipse.org/projects/modeling.eatop). This API facilitates the creation and manipulation of the eaxml files by reason of methods for creating EAST-ADL2 entities or setting of their properties, etc. The following video shows an example of serialization of the EAST-ADL2 model in eaxml file using EAXmlGen - http://www.youtube.com/watch?v=0Y2Uk6rcCiA.
- EAQompGateway – generates an analysis context (expressed with the SysML and MARTE) from the EAST-ADL2 model. This analysis context is managed by the Qompass framework. The generation was done purely in Java with no additional frameworks such as QVT. The EAST-ADL2 model taken as an input for the generation refers to the *Functional Design Architecture* viewpoint, *Hardware Architecture* viewpoint, *Allocation* viewpoint and *Timing* viewpoint. The model developed under the

Functional Design Architecture viewpoint serves to generate the workload behavior, i.e. using employed notation this means control flows of the activity diagram. The hardware viewpoint is used to generate the resources platform. The allocation viewpoint is a deployment specification but of course at the EAST-ADL2 level it refers only to the specification of allocation of atomic design functions to the hardware. If the allocation information is not present the transformation will produce a partial analysis context, subject for optimization. Finally the timing viewpoint delivers the timing properties and constraints (execution times, deadlines) indispensable to run the analysis. The following video shows generation of the analysis context from the EAST-ADL2 using EAQompGateway - http://www.youtube.com/watch?v=vyEVCDEKl2Y. It also shows an example of analysis, e.g. utilization of ECUs computed for particular allocation configuration.

- ARQompGateway – generates an analysis context or similarly a partial analysis context from the AUTOSAR model. The AUTOSAR model taken as an input for the generation refers to the models developed under all the viewpoints of the Technical Layer except the technical layer exchange viewpoint. The principle of transformation is similar as for the EAQompGateway. Implementation is using a pure Java.
- ARXmlGen – generates arxml file from the AUTOSAR model created in Papyrus with its UML profile. Its purpose is to serialize and deserialize the AUTOSAR model within an arxml file. The last is an xml file whose content conform to the xml schema of the corresponding AUTOSAR release. The arxml format is supported by the Artop (AUTOSAR Tool Platform). This platform is an implementation of a common base functionality for AUTOSAR development tools inter alia modeling of AUTOSAR architectures. It also provides Java API to manipulate the arxml files which was used to create and manipulate arxml files based on the AUTOSAR model created in Papyrus. The following video shows an example of the generation of arxml file out of the Papyrus AUTOSAR model using ARXmlGen - http://www.youtube.com/watch?v=qmHW0LOzpjw.
- ARGateway – generates AUTOSAR model from the EAST-ADL2 model. It was implemented as a Java transformation. The ARGateway employs the strategy in which each runnable entity is generated from one atomic function. Concerning their composition

in software components, it follows the compositional structure of the EAST-ADL2 model. For instance if two atomic functions are in the same composite design function, they will be put in the same software component. This strategy can be overwritten with the specification of a generation strategy using the proposed UML profile. In that case, ARGateway for those atomic functions for which generation strategy was explicitly specified using the profile, will overwrite its default strategy. The operation of the ARGateway is shown in the following video - http://www.youtube.com/watch?v=qmHW0LOzpjw.

Analysis and Optimization Algorithms

The analysis and optimization algorithms, similarly as the transformations, were implemented as an Eclipse plugins. In order to run the analysis user needs to select the analysis context of interest, make a right click to display the menu, then choose the Qompass framework, and appropriate action. The Figure A.6 shows an example of running the Offset-based Schedulability Analysis [40] on one of the defined analysis contexts. The analysis, no matter which one, will traverse the model of selected analysis context to extract the information necessary for particular analysis algorithm. For instance if this is response times analysis, information such as host demand of behavior actions stereotyped with the *SaStep* will be considered.

The optimization works in a similar way however in this case the user needs to select the model representing optimization context and then choose an appropriate optimization technique. The Figure A.7 shows an example for running the optimization concerning time budgets assignment, taking as an input one of the optimization contexts.

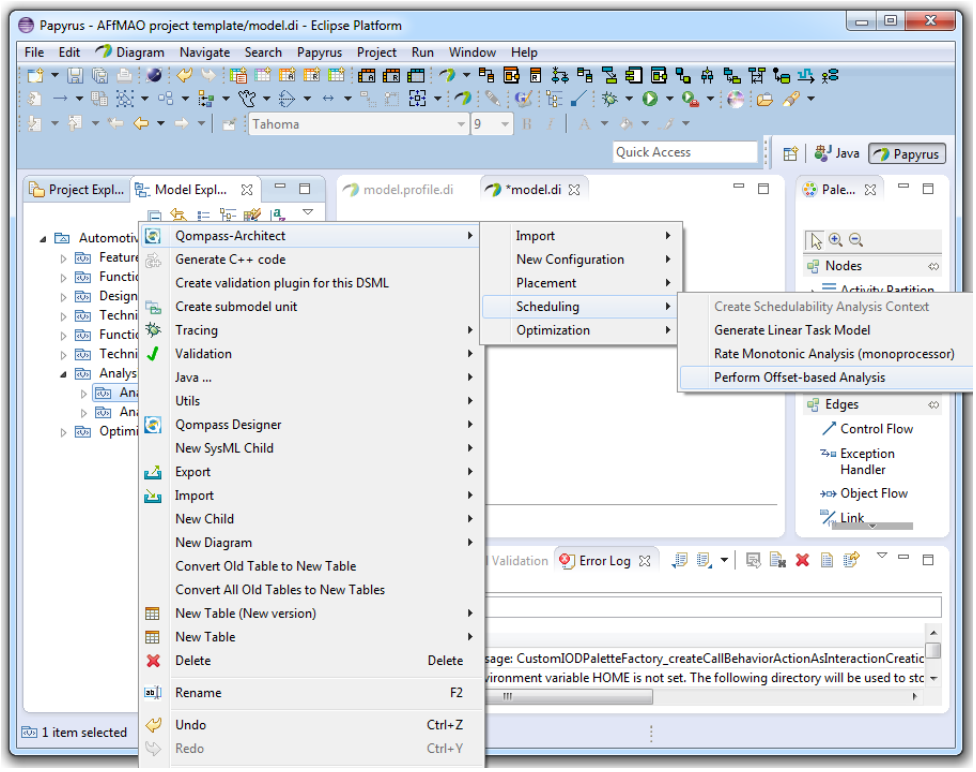

Figure A.6. Running Schedulability Analysis on the Analysis Context

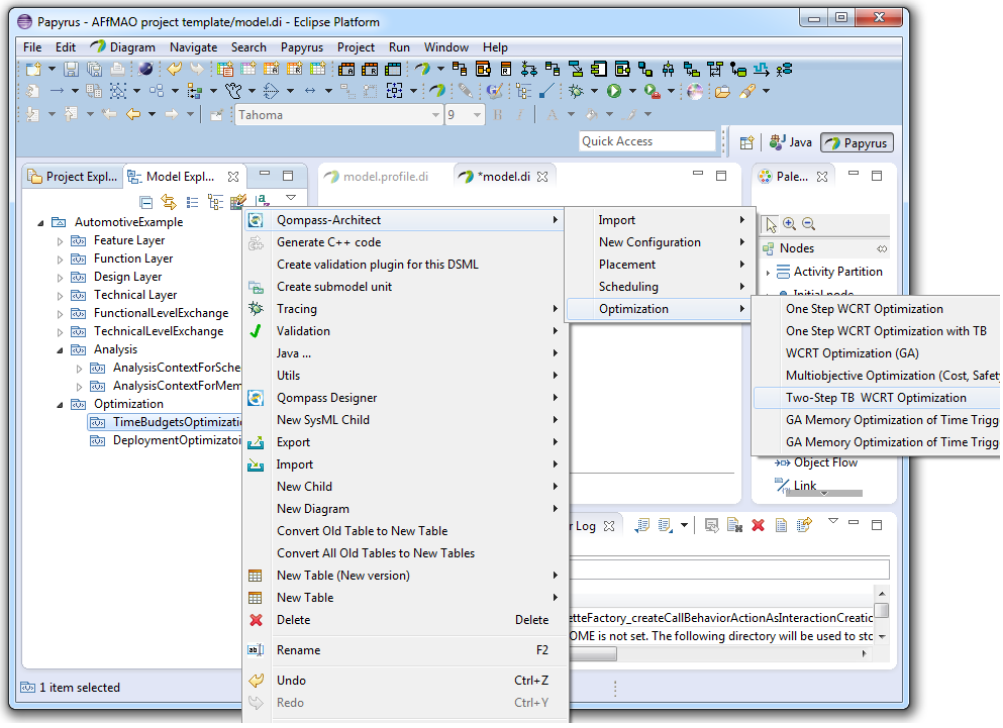

Figure A.7. Running Optimization (Time Budgets Assignment) on the Optimization Context

B. AAF Revisited

The table below lists all the viewpoints presented in the chapter 5. It discusses their main properties, i.e. the concerns and model kinds. The last is composed of three properties: *language* used, *modeling technique* (mechanism such as UML profile plus the used diagram) and *analytical methods* such as algorithms for the analysis or optimization if applicable.

Architecture Viewpoint	**Concern**		**Model Kind**		
			Language	**Modeling Technique**	**Analytical Methods**
Feature Analysis Architecture (FAA)	Specification of vehicle features		Subset of EAST-ADL2	UML profile/UML Composite diagram	
Functional Analysis Architecture (FunAA)	Specification of function types		Subset of EAST-ADL2	UML profile/UML Composite diagram	
	Specification of function prototypes		Subset of EAST-ADL2	UML profile/UML Composite diagram	
Functional Design Architecture (FDA)	Refinement of function types specified at functional layer		Subset of EAST-ADL2	UML profile/UML Composite diagram	
	Refinement of function prototypes specified at functional layer		Subset of EAST-ADL2	UML profile/UML Composite diagram	
Hardware Architecture (HA)	Provision of abstract hardware platform specification	Types of hw elements	Subset of EAST-ADL2	UML profile/UML Composite diagram	
		Prototypes of hw elements	Subset of EAST-ADL2	UML profile/UML Composite diagram	
Allocation	Allocate entities from FDA to those from HA		Subset of EAST-ADL2	UML profile/UML Composite diagram	
Timing	Provision of timing characteristics at the feature, function and design layer		Subset of EAST-ADL2 for timing (TADL)	UML profile/diagrams of FAA, FunAA and FDA	

				extended with timing information	
Application	Specification of software architecture	Types of software entities	Subset of AUTOSAR	UML profile/UML Class diagram	
		Prototypes of software entities	Subset of AUTOSAR	UML profile/UML Composite diagram	
Topology	Provision of hardware platform specification	Types of hw elements	Subset of AUTOSAR	UML profile/UML Composite diagram	
		Prototypes of hw elements	Subset of AUTOSAR	UML profile/UML Composite diagram	
Internal Behavior	Behavioral decomposition of software components		Subset of AUTOSAR	UML profile/UML Activity diagram	
Application Allocation	Allocate entities from Application viewpoint to those from the Topology viewpoint		Subset of AUTOSAR	UML profile/UML Composite diagram	
ECU Configuration	Completion of the deployment specification by formulating the tasks, specifying runnables partitioning and priorities assignment		Subset of AUTOSAR	UML profile/UML Class diagram	
Application Timing	Specification of timing information for the software architecture		AUTOSAR Timing Extensions	UML profile/UML Composite diagram	
Generation	Specification of strategy for the generation of preliminary implementation model out of the functional model		Metamodel for Generation Strategy	UML profile/UML Composite diagram	

Analysis	Enabling analysis of automotive architectures in a design process	Subset of SysML and MARTE	UML4SysML Activity diagram	Algorithms for response times analysis and memory overhead computation
Optimization	Enabling optimization of automotive architectures in a design process	Subset of SysML, MARTE and metamodel for optimization	UML profile for optimization metamodel/UML 4SysML Activity diagram	Techniques for design space exploration

Table B.0.1. Viewpoints of the AAF with their Concerns and Model Kinds